André Tiago Marques Malafaia

Licenciado em Engenharia Informática

Sistema de Informação para Projetos de Monitoração Fitossanitária

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: Carlos Viegas Damásio, Professor Associado, Universidade Nova de Lisboa

Co-orientadores: João Moura Pires, Professor Auxiliar, Universidade Nova de Lisboa

Armanda Rodrigues, Professora Auxiliar, Universidade Nova de Lisboa

Júri

Presidente:	Doutora Ana Maria Diniz Moreira
Arguente:	Doutor Luís Miguel Mendonça Rato
Vogal:	Doutor Carlos Viegas Damásio

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2019

Sistema de Informação para Projetos de Monitoração Fitossanitária

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Agradecimentos

Ao meu orientador Carlos Damásio e ao meu co-orientador João Moura Pires pelo acompanhamento exemplar que me deram ao longo da elaboração da minha dissertação, por todas as horas que estivemos em reunião, por toda a ajuda prestada, por me manterem motivado em fazer sempre mais e melhor. Contribuíram não só para a minha dissertação mas também para o meu crescimento pessoal e profissional.

À minha co-orientadora Armanda Rodrigues que se mostrou sempre disponível para me ajudar no que fosse necessário.

A toda a equipa do projeto FitoAgro (PDR2020-101-031686) por toda a colaboração ao longo de todos os meses de trabalho.

Aos meus pais e irmão - Ilídio, Helena e Pedro - por todo o apoio que me deram desde sempre. À minha mãe e ao meu pai por todos estes anos de esforço e dedicação. Ao meu irmão por todo o interesse demonstrado relativamente ao trabalho realizado.

À minha namorada Vera pelo apoio incondicional e por toda a força e motivação que me dá em tudo o que faço. Por todas as vezes que tentaste perceber o idioma 'chinês' quando falava contigo sobre o meu trabalho e pela paciência de ler todo este documento. Sem ti nada seria igual.

Ao meu amigo e colega Diogo Albuquerque, com quem partilhei todos os altos e baixos da jornada que foi este curso e que ao longo deste trabalho sempre se mostrou interessado e disponível para dar as suas opiniões.

Por fim, ao meu gato Bit pela companhia que me fez cada vez que trabalhei em casa, especialmente por todas as vezes que saltou para cima da minha secretária e se deitou em cima do teclado enquanto eu trabalhava ou escrevia.

A todos,
O meu muito obrigado!

Data! Data! Data! I can't make bricks without clay.

The Adventure of the Copper Beeches
Sir Arthur Conan Doyle

Resumo

Ao longo dos tempos, as pragas têm sido uma adversidade para os produtores agrícolas. Capazes de dizimar culturas inteiras e causar prejuízos económicos enormes, torna-se assim necessária a aposta no controlo, na prevenção e na deteção das mesmas.

Atualmente a produção hortofrutícola do Oeste depara-se com vários problemas de índole fitossanitária que precisam de soluções inovadoras. As alterações climáticas têm criado condições favoráveis ao aumento das populações de pragas, à alteração de ciclos de vida das mesmas e à ocorrência de novos inimigos, que até então eram desconhecidos ou tinham pouca importância e que se tornaram economicamente relevantes.

Esta dissertação é desenvolvida no âmbito do projeto FitoAgro, projeto esse que tem como objetivo estudar as pragas emergentes que afetam as pomóideas do Oeste, contribuindo para o ajustamento de metodologias de monitorização, para a definição de estimativas de risco e níveis económicos de ataque para estes novos inimigos e também para definir uma estratégia de controlo regional para prevenir e minimizar os problemas causados por esses mesmos inimigos.

O principal propósito desta dissertação centra-se no desenho de um sistema de base de informação genérico que seja aplicável a projetos deste domínio e na sua implementação para o projeto FitoAgro. Trata-se de um sistema de informação georreferenciada que irá acomodar dados meteorológicos, quer estes venham de estações meteorológicas ou de previsões, e ainda qualquer tipo de recolha de informação que seja efetuada em campo.

Palavras-chave: Sistema de informação, sistema de recolha de dados, dados meteorológicos, dados recolhidos em campo, sistema de informação geográfica, pragas de culturas.

Abstract

Throughtout times, pests have been an adversity to the agricultural producers. Capable of decimate entire cultures and to cause massive economic losses, its necessary to bet on the control, prevention and detection of pests.

Nowadays the horticultural production of the West region comes across with several problems of phytosanitary nature that need innovative solutions. The climate change has created favorable conditions to the increase of the populations of pests, to the change of their life cycles and the occurence of new enemies, that were so far unknown or held little importance but have become economically relevant.

This dissertation is developed under the FitoAgro project, that has the goal of studying emergent pests that affect the fruit trees in the West region, contributing to the adjustment of monitoring methodologies, to the definition of risk estimates and economic levels of attack of this new enemies and also to define a regional control strategy to prevent and minimize the problems caused by the enemies.

The main purpose of this dissertation focuses on the design of a generic information base system that is applicable to projects of this domain and its implementation for the FitoAgro project. Its a geographic information system that will accommodate meteorological data whether it comes from meteorological stations or from predictions and also any type of information gathering done in the field.

Keywords: Information system, data collection system, meteorological data, field data, geographic information system, crop pests.

Índice

Lista de Figuras

Lista de Tabelas

Glossário

diapausa	Retenção temporária do desenvolvimento.
eager-loading	Tipo de load onde modelos relacionadas são carregadas na mesma query.
fitossanitária	Que diz respeito ao estado de saúde das espécies vegetais.
hortofrutícola	Que produz ou comercializa produtos hortícolas e frutos simultaneamente.
lazy-loading	Tipo de load onde apenas o modelo consultado é carregado.
níveis económicos de ataque	Intensidade de ataque de um inimigo da cultura a que se devem aplicar medidas limitativas ou de combate para impedir que a cultura corra riscos de prejuízos superiores ao custo das medidas de luta a adotar, acrescido dos efeitos indesejáveis que estas últimas podem provocar.
parcela	Constitui a porção contínua de terreno homogéneo com limites estáveis agronómica e geograficamente, com uma identificação única conforme registado no Sistema de Identificação Parcelar.
pomóideas	Designação das árvores e arbustos cujos frutos comestíveis são pomos como a maçã e a pêra.
pontos de observação biológica	Parcelas que têm o objetivo de monitorizar um ou mais problemas fitossanitários.

pragas	Surto de insectos nocivos ao desenvolvimento agrícola.
produtos fitofarmacêuticos	Produtos químicos utilizados para combater e evitar pragas e doenças agrícolas, protegendo as plantas e as culturas.
região do Oeste	Região delimitada a sul pela Grande Lisboa, a leste pela Lezíria do Tejo, a norte pelo Pinhal Litoral e a oeste pelo Oceano Atlântico.

Siglas

CRUD	Create, Read, Update, Delete.
FTP	File Transfer Protocol.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
IPMA	Instituto Português do Mar e da Atmosfera.
LST	Land surface temperature.
NEA	Níveis económicos de ataque.
OMM	Organização Meteorológica Mundial.
ORM	Object-Relational Mapping.
POB	Ponto de observação biológica.
SFTP	Secure File Transfer Protocol.
WFS	Web Feature Service.
WMS	Web Map Service.

Capítulo 1

Introdução

Este capítulo tem como objetivo, numa fase inicial, descrever o contexto e a motivação (1.1) desta dissertação. De seguida é apresentada uma descrição do problema (1.3), prosseguida pelos objetivos pretendidos e pelas contribuições esperadas (1.4). Para concluir, são apresentados os capítulos (1.5) deste documento, acompanhados por uma breve descrição.

1.1 Contexto e Motivação

Ao longo dos tempos, as pragas têm causado inúmeros problemas à humanidade. Capazes de dizimar culturas inteiras, foram responsáveis por períodos de fome e pela extinção de recursos naturais, tendo assim um claro impacto económico e social [38].

Nos dias de hoje, apesar das pragas normalmente não apresentarem consequências tão nefastas como no passado, apresentam problemas para os agricultores resultando por vezes em graves perdas económicas, acabando também por reduzir a qualidade do produto final que é apresentado ao consumidor.

Para reduzir estas consequências, os produtores agrícolas tentam prevenir os surtos de insetos utilizando técnicas de controlo de pragas, recorrendo principalmente ao uso de produtos fitofarmacêuticos caso essa prevenção tenha falhado [38]. A prevenção tem sido a aposta dos agricultores visto que evita perdas nas culturas e para além disso as técnicas de prevenção costumam ser menos dispendiosas para os produtores e também menos prejudiciais para o consumidor e para o ambiente quando comparadas com a aplicação de produtos químicos.

Desse modo, têm surgido diversos projetos de investigação cujo objetivo tem sido estudar essas pragas de modo a que a prevenção se consiga fazer de forma mais eficiente. Todos esses projetos têm sentido a necessidade de ter um sistema de informação que suporte toda a informação gerada por estes e que a disponibilize para os seus estudos. No entanto,

cada vez que surge um projeto novo tem-se implementado de raiz o tão necessitado sistema de informação - não existe uma implementação genérica que permita ser reutilizada.

Portugal não tem sido exceção. A produção hortofrutícola da região do Oeste depara-se com vários problemas de índole fitossanitária, novas pragas têm proliferado e assumem hoje uma grande importância para os produtores visto que apresentam uma crescente relevância económica para as suas atividades. Nesse contexto surge o projeto FitoAgro que pretende definir uma estratégia de controlo regional para prevenir e minimizar os problemas causados pelas pragas emergentes no Oeste dando assim resposta às necessidades dos agricultores. Associado a este projeto surgiu então a necessidade de criar um sistema de informação e apoio para técnicos e agricultores no âmbito da fitossanidade de maneira a que estes possam cumprir as suas obrigações legais e ainda a necessidade de informação e registo regional de pragas, essencial para a elaboração de planos de risco fitossanitário, na abertura de novos mercados de exportação.

Assim sendo, com esta dissertação é pretendido desenhar e implementar um sistema de informação genérico para projetos deste âmbito, tendo como o seu caso de estudo o projeto FitoAgro.

1.2 Projeto FitoAgro

O projeto FitoAgro é um projeto nacional cujo objetivo passa pelo estudo dos ciclos de vida das novas pragas que afetam as pomóideas, como as pereiras e macieiras, da região do Oeste, de maneira a que se possam definir metodologias de estimativa de risco e os respetivos níveis económicos de ataque (NEA), validar modelos de desenvolvimento de pragas e construir mapas de risco. Pretende-se com base nesse estudo ensaiar métodos de luta biotécnica e biológica de forma a precisar estratégias de controlo baseadas nesses métodos, contribuindo para a diminuição do recurso a produtos químicos e consequentemente reduzindo os riscos para a saúde humana e para o ambiente. É também esperada a introdução do uso de dispositivos móveis como meios de registo de ocorrências, servindo-se das capacidades de georreferênciação e anexação de fotos para o aumento do rigor na documentação das pragas.

Este projeto [18] é financiado através do "PDR2020 - Programa de Desenvolvimento Rural" no âmbito do acordo Portugal 2020, contando com a contribuição de diversas instituições, nomeadamente:

- COTHN - Centro Operativo e Tecnológico Hortofrutícola Nacional;
- FCT/UNL - Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa;
- ISA - Instituto Superior de Agronomia;
- APAS - Associação dos Produtores Agrícolas da Sobrena;
- ESACB - Escola Superior Agrária de Castelo Branco;

- ESAS - Escola Superior Agrária de Santarém;
- Frutus - Estação Fruteira do Montejunto CRL;
- Frutoeste - Cooperativa Agrícola de Hortofruticultores do Oeste CRL;
- CPF - Centro de Produção e Comercialização Hortofrutícola Lda;
- CAB - Cooperativa Agrícola do Bombarral CRL;
- Granfer - Produtores de Frutas CRL;
- COOPVAL - Cooperativa Agrícola dos Fruticultores do Cadaval CRL;
- Ecofrutas - Estação Fruteira da Estremadura Lda;
- Frubaça - Cooperativa de Hortofruticultores CRL.

Os parceiros hortofrutícolas disponibilizaram parcelas para servirem de pontos de observação biológica (POB) nos quais foram instalados vários dispositivos de monitorização (armadilhas cromotrópicas, armadilhas sexuais, cintas, etc.) que permitirão o estudo e validação dos ciclos de vida de maneira a definir-se a tão necessária estratégia de controlo regional com principal foco na luta biotécnica e biológica.

1.2.1 Problemas nas pomóideas do Oeste

Este projeto foca-se no estudo de quatro espécies de insetos que têm vindo a afetar de forma crescente a produção das pomóideas do Oeste. Visto as pomóideas terem uma única produção por ano, o efeito destas espécies pode arruinar o retorno do trabalho anual realizado pelos agricultores. As quatro espécies em estudo são:

- Filoxera (*Aphanostigma pyri* Chol.)
- Cecidómia (*Dasineura pyri* Bouché)
- Novo Lepidóptero (*Cydia pomonella* L.)
- Cochonilha-Algodão (*Pseudococcus viburni*)

De seguida será apresentada uma breve descrição de cada espécie, cujas características permitem explicar a necessidade de controlo regional.

1.2.1.1 Filoxera

A Filoxera é um pequeno homóptero (entre 0,8 a 1 mm de comprimento) que tem cinco a oito gerações por ano [17]. Passa o inverno em diapausa sob a forma de ovos, depositados normalmente nas reentrâncias das cascas das árvores. Esses ovos eclodem entre os meses de março e abril sendo que as larvas resultantes destes desenvolvem-se consoante as condições

climáticas originando fêmeas fundadoras. Surge assim a primeira geração de Filoxera do ano [16]. Na figura 1.1 é possível ver uma colónia desta praga.

Cada fêmea desta espécie pode depositar entre 60 a 100 ovos e as larvas apresentam alta mobilidade o que permite que alcancem facilmente o seu alimento: os frutos. Normalmente em setembro dá-se a última geração do ano.

Devido a não possuir asas, esta espécie não migra para outras árvores, devendo-se assim a sua dispersão à ação do homem.

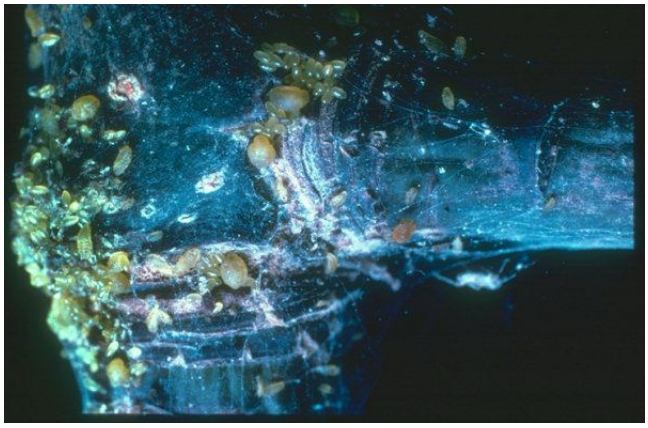


Figura 1.1: Colónia de Filoxera na primavera (fêmeas, ovos e larvas) [16]

1.2.1.2 Cecidómia

A Cecidómia é um pequeno díptero, cujo adulto atinge entre 1,5 a 2 mm de comprimento. Tem normalmente duas a quatro gerações por ano, sendo a última em julho ou agosto, altura em que as larvas se enterram no solo e entram em diapausa até à primavera do ano seguinte [6].

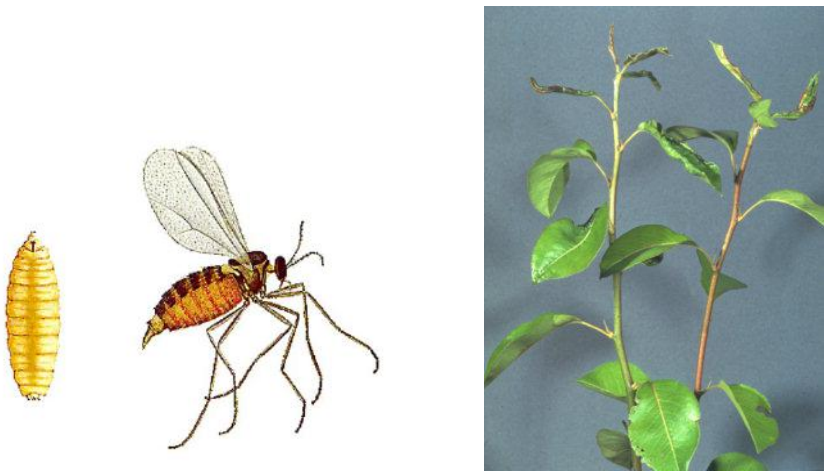
Os danos causados por esta espécie incidem diretamente nas folhas uma vez que é esse local onde são depositados os ovos e onde as larvas se alimentam. As folhas atacadas encontram-se enroladas (1.2b) longitudinalmente, com larvas esbranquiçadas no seu interior. As folhas tornam-se quebradiças por dessecação, verificando-se em ataques muito graves a desfoliação dos lançamentos e a paragem do crescimento vegetativo. Apesar das árvores adultas não sofrerem tanto com estes danos, as árvores jovens sofrem bastante uma vez que os ataques desta praga dificultam o seu crescimento.

Devido a possuir asas (1.2a), é um inseto que facilmente se espalha de pomar em pomar.

1.2.1.3 Novo Lepidóptero

O Novo Lepidóptero é um lepidóptero que apresenta duas gerações anuais em Portugal. Passa o outono e o inverno em diapausa, escondido normalmente sob a casca das árvores na forma de larva [44].

Dependendo das condições climáticas, as larvas começam o seu desenvolvimento em abril, dando origem à primeira geração de insetos adultos - borboletas (1.3a). Cada fêmea

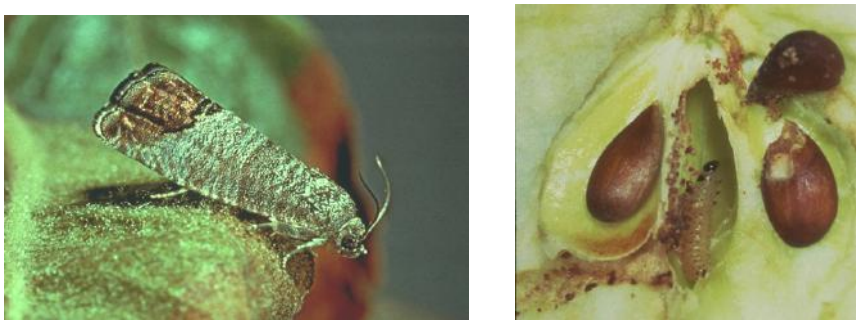


(a) Pupa e fêmea adulta de Cecidómia (b) Folhas enroladas

Figura 1.2: Ilustração de cecidómia e ataque a folhas [6]

põe cerca de 60 ovos por outros tantos frutos. Ao eclodirem dos ovos, as larvas deste inseto penetram os frutos, alimentando-se do seu conteúdo (1.3b) e desvalorizando-os comercialmente [8].

Esta praga apresenta uma grande mobilidade devido aos adultos serem borboletas e os prejuízos em pomares não tratados podem atingir os 90% da produção.



(a) Adulto de Novo Lepidóptero (b) Larva devorando o fruto

Figura 1.3: Espécime adulto e ataque no fruto [44]

1.2.1.4 Cochonilha-Algodão

A Cochonilha-Algodão é um hemiptero cujo número de gerações anuais varia entre três e cinco. Passa todo o inverno em diapausa sob a forma de ovos, ninfas ou adultos, escondidos comumente nas bases dos troncos. Na primavera, com a subida da temperatura, voltam novamente à atividade e cada fêmea pode produzir até 250 ovos, devendo a sua dispersão para outras árvores à ação do vento, de aves ou de mamíferos [39].

O estrago causado por esta espécie tem como base a alimentação da seiva do hospedeiro e a substância que excreta com a sua atividade (1.4), desvalorizando comercialmente os frutos e podendo facilmente resultar em prejuízos significativos [5].

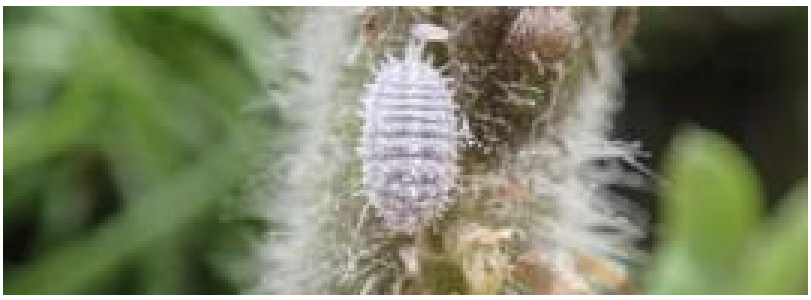


Figura 1.4: Cochonilha-Algodão e substância excretada [5]

1.2.2 Modelos de previsão

Tal como foi referido em 1.2, o projeto FitoAgro pretende desenvolver e confirmar modelos de previsão para os inimigos enunciados em 1.2.1. Como é explicado em [32], o desenvolvimento dos insetos está diretamente ligado às condições meteorológicas do local onde se encontram. Existe um intervalo de temperaturas em que os insetos se conseguem desenvolver, acima e abaixo do qual têm o seu desenvolvimento suspenso. É neste ponto que o cálculo de graus-dia acumulados é útil. Um grau-dia corresponde a um dia (24 horas) onde a temperatura se encontra um grau acima do limite inferior de desenvolvimento (e abaixo do máximo caso exista). A acumulação deste valor ao longo dos dias permite estimar a altura do ciclo de vida em que o inseto se encontra [32], informação muito importante para o sucesso dos métodos de luta biotécnica e biológica.

Deste modo, o desenvolvimento, a validação e, mais tarde, a execução destes modelos está completamente dependente da existência de diversa informação com qualidade. Para que se possa ter uma noção, um modelo para ser validado requer pelo menos 10 anos de informação de qualidade. A informação necessária para estes modelos é descrita de seguida.

1.2.3 Informação meteorológica

No âmbito deste projeto instalaram-se diversas estações meteorológicas, por norma perto dos POBs, para se proceder ao registo de dados meteorológicos referentes a locais próximos de cada POB. Para além das estações, o projeto terá outras fontes deste tipo de dados, nomeadamente satélites, previsões meteorológicas e fornecedores online. Estes dados assumem particular importância devido a serem a base dos modelos de previsão de pragas, como foi referido em 1.2.2.

Torna-se necessário ter a todo o momento informação meteorológica disponível e com qualidade, tanto para na fase inicial do projeto se proceder ao estudo e à validação dos ciclos de vida das pragas emergentes bem como para no futuro se executarem os modelos de previsão.

1.2.4 Recolha de informação em campo

Os POBs são visitados semanalmente por técnicos que registam o que observam nos diversos dispositivos de monitorização que lá se encontram instalados. Cada inimigo tem um protocolo de recolha de informação associado e os técnicos fazem o registo dos dados segundo esses protocolos. Os dados referidos são particularmente relevantes pois permitem avaliar a incidência de organismos nocivos ao longo do tempo. Atualmente os dados recolhidos em campo são apenas disponibilizados através de um Google Drive criado no âmbito deste projeto.

No entanto, muitas vezes, a informação não se encontra normalizada o que constitui um problema para a parametrização de modelos. Assim sendo, é necessário tratar este tipo de dados de forma o mais genérica possível, oferecendo ainda suporte para a validação e visualização dos mesmos.

1.2.5 Informação georreferenciada

Existe a necessidade de georreferenciação de todos os POBs bem como de toda a informação anteriormente descrita de forma a que se possam gerar mapas de risco.

1.3 Descrição do Problema

O problema que esta dissertação pretende resolver é o desenvolvimento de um sistema de informação reutilizável para os projetos de monitorização fitossanitária típicos. Este tem várias necessidades apresentadas de seguida.

1.3.1 Âmbito Informacional

O âmbito do sistema de informação prende-se com a fitossanidade. Desse modo deverá acomodar dados como os que são gerados pelo projeto FitoAgro: dados meteorológicos, dados de campo, representação dos protocolos de recolha de dados no campo, georreferenciação de parcelas e dos dados gerados.

1.3.2 Requisitos Funcionais

O sistema deve tratar de duas partes distintas: a parte operacional e a parte analítica. A parte operacional diz respeito à disponibilização e integração dos dados gerados pelos projetos, assegurando a qualidade dos mesmos. A parte analítica corresponde à capacidade do sistema proporcionar mecanismos que suportem as necessidades analíticas dos projetos.

1.3.3 Requisitos Técnicos e Tecnológicos

Não existem quaisquer restrições para a parte técnica e tecnológica do sistema a implementar, havendo total liberdade de escolha no que toca a este assunto.

1.3.4 Requisitos de Generalização

O sistema deve ter um carácter genérico permitindo a sua reutilização para os mais diversos projetos de âmbito fitossanitário bem como a sua adaptação aos aspetos específicos de cada um.

1.3.5 Requisitos de Usabilidade

O sistema devará ser fácil de aplicar e de adaptar aos projetos deste âmbito. De forma igual também deverá ser fácil de utilizar pelos envolvidos nos projetos.

1.3.6 Requisitos de Desempenho e Escalabilidade

O sistema deverá ser escalável, permitindo a sua aplicação a projetos com dimensões diferentes. O desempenho do sistema não tem qualquer requisito imposto mas deverá ser no mínimo comparável ao desempenho dos sistemas semelhantes desenvolvidos no passado.

1.4 Objetivos e Contribuições

O grande objetivo desta dissertação é, tal como já foi referido anteriormente, a modelação e implementação de um sistema de base de informação genérico, aplicando-o ao caso de estudo FitoAgro.

Este sistema deverá proceder à centralização e agregação de toda a informação meteorológica proveniente de várias fontes, quer sejam as estações meteorológicas do projeto, dados de satélite, dados de previsão ou dados de fornecedores online. Toda esta informação deverá ser limpa, curada, ordenada e processada para que possa ser integrada em modelos de previsão que serão executados no futuro neste sistema.

Deverá ainda acomodar todos os dados provenientes dos protocolos de monitorização de inimigos nos POBs, apresentando um carácter extensível de maneira a que novos protocolos, novos inimigos e novos modelos de previsão possam ser adicionados.

Outro ponto importante é que deverá georreferenciar POBs, a informação resultante da monitorização feita nestes e ainda a informação meteorológica.

Por fim, deverá ter em conta, no âmbito do projeto FitoAgro, a futura integração com a plataforma InfoAgro da COTHN que, apesar de já ter modelos de previsão de pragas e doenças e alguns mapas de risco, ainda não os possui para os inimigos emergentes que se encontram a ser estudados neste projeto.

Esta dissertação pretende então contribuir a curto prazo para o sucesso do projeto FitoAgro e para a resolução dos problemas que o Oeste atravessa com a aplicação do sistema de informação desenvolvido, que permitirá a este projeto atingir os seus objetivos: sendo numa fase inicial o estudo das novas pragas e a validação dos modelos para esses inimigos e numa segunda fase a aplicação desses modelos, a construção de mapas de risco e o ensaio do uso das técnicas de luta biológica e biotécnica. Assim será possível um combate às pragas

mais eficiente e mais amigo do ambiente, aprimorando a fitossanidade das pomóideas do Oeste e consequentemente melhorando a qualidade e a segurança alimentar dos produtos que delas resultam.

Olhando mais a longo prazo, pretende-se aplicar o sistema resultante desta dissertação aos projetos futuros desta área.

1.5 Estrutura do Documento

Este documento encontra-se dividido em seis capítulos e está estruturado da seguinte forma:

- **Introdução**

  Este capítulo (1) contém uma introdução ao tema a ser desenvolvido por esta dissertação que expõe os objetivos e motivação do trabalho a ser realizado.

- **Trabalho Relacionado**

  O capítulo 2 tem presente o trabalho de pesquisa efetuado durante o período de preparação da dissertação. Aqui foi realizado o levantamento de standards, tecnologias, ferramentas e aplicações que foram consideradas relevantes para o nosso problema.

- **Modelação**

  O capítulo 3 compreende uma análise de alto nível da solução implementada, explicitando o processo de modelação.

- **Implementação**

  Em 4 são apresentadas e discutidas partes específicas da implementação do sistema de informação.

- **Avaliação**

  O penúltimo capítulo 5 é dedicado à avaliação da solução desenvolvida, incluindo uma discussão sobre a aplicabilidade do sistema de informação a projetos do mesmo domínio e um conjunto de testes de desempenho.

- **Conclusões**

  Por fim, o capítulo 6 terá as conclusões finais referentes a esta dissertação bem como um conjunto de melhorias ao sistema de informação que poderão vir a ser desenvolvidas no futuro.

Capítulo 2

Trabalho Relacionado

Neste segundo capítulo é apresentado o trabalho relacionado relevante para esta dissertação. Primeiramente são apresentadas aplicações com funcionalidades semelhantes ao sistema que se pretende desenvolver (2.1), seguido de um estudo no âmbito de informação geográfica (2.2), dados meteorológicos (2.3), técnicas de interpolação espacial (2.4) e dados de satélite (2.5). Para fechar o capítulo são descritas diversas tecnologias (2.6) consideradas relevantes para esta dissertação.

2.1 Projetos Relacionados

Esta primeira secção tem o objetivo de fazer o levantamento de aplicações cujo domínio de ação é igual ao do sistema que se pretende desenvolver no âmbito do projeto FitoAgro, ou seja, aplicações do domínio agrícola que pretendem em última análise auxiliar os agricultores a maximizar a sua produção.

2.1.1 Protomate

O projeto Protomate [34] resultou no desenvolvimento de um sistema que obtém dados meteorológicos provenientes de satélite e de estações meteorológicas da rede do Instituto Português do Mar e da Atmosfera (IPMA), utilizando-os para a execução de modelos de previsão de doenças da vinha e dos tomateiros. Os utilizadores interagiam com o sistema através de uma aplicação móvel que permitia que introduzissem as suas parcelas, indicando os detalhes associados a estas, registassem observações de doenças e tratamentos realizados, recebendo ainda alertas consoante o resultado da execução dos diversos modelos de previsão.

2.1.2 Safebrócolo

O projeto Safebrócolo [48] culminou no desenvolvimento de uma plataforma que permite a recolha de informação de campo georreferenciada através de uma aplicação móvel. Essa recolha de informação permitiu o desenvolvimento e a validação de modelos de risco para dois problemas que afetam a cultura do bróculo em Portugal - o fungo *Alternaria brassicola* e a mosca *Delia radicum*. A plataforma é única e exclusivamente orientada a esses dois problemas e atualmente permite aos utilizadores que registem as suas parcelas, que documentem as observações efetuadas no campo através da aplicação móvel e ainda lhes disponibiliza tanto um mapa de risco para Portugal Continental como um indicador de risco para cada uma das suas parcelas.

2.1.3 Cropio

O Cropio [9] é um sistema de gestão agrícola por satélite que facilita a monitorização remota de parcelas, permitindo que os seus utilizadores planeiem e executem as suas operações agrícolas de forma eficiente, maximizando a produtividade do seu negócio. Ao usar imagens de satélite em tempo real, previsões meteorológicas precisas e outras informações críticas, o sistema permite que os utilizadores possuam informação do estado da sua cultura, dos níveis de vegetação e até dos locais onde é mais provável virem a ocorrer problemas. Adicionalmente, entrega semanalmente um sumário de dados relevantes sobre a condição das parcelas monitorizadas. Isto permite que haja um acompanhamento contínuo do crescimento da cultura desde que é plantada até ao momento da colheita como é possível ver na figura 2.1.

Este sistema está disponível em diversas línguas e o seu acesso é pago, sendo exigido aos utilizadores entre 1 a 5 dólares por hectare por ano.

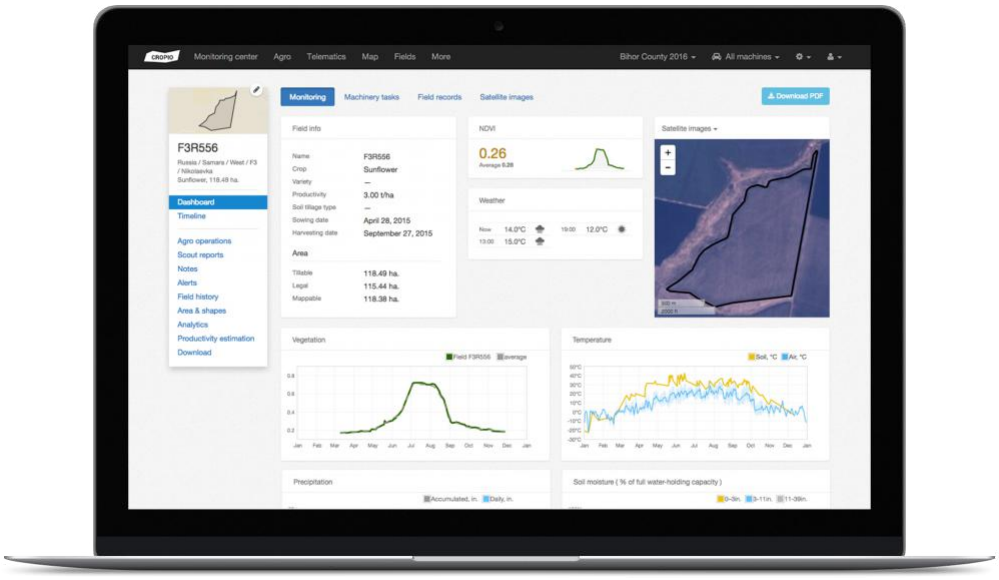


Figura 2.1: Vista geral de uma parcela no sistema Cropio [9]

2.1.4 WiseCrop

A WiseCrop [61] é uma *startup* portuguesa que oferece um sistema de apoio à gestão agrícola que ajuda o produtor a poupar recursos e a aumentar a produção. Fornece ao empresário agrícola dados, quase em tempo real (atualização de 15 em 15 minutos), sobre o estado das suas culturas através da medição de vários parâmetros meteorológicos como a temperatura, humidade, velocidade e direção do vento, radiação, humidade do solo, entre outros. Essa medição é feita a partir de diversos sensores que são colocados nas parcelas que os utilizadores pretendem monitorizar.

Associado a essas medições, possui um sistema de alertas que utiliza modelos de previsão para avisar o produtor sobre a ocorrência de pragas, doenças ou variações de clima que possam prejudicar a cultura. Apresenta ainda um carácter personalizável visto dar como opção ao utilizador a definição dos seus próprios alertas.

Este sistema é pago anualmente, sendo adquirido por módulos consoante as necessidades do utilizador.

2.2 Informação Geográfica

O facto do sistema de informação, a ser desenvolvido nesta dissertação, possuir uma forte componente geográfica motivou o estudo do estado da arte no que toca à modelação e à partilha de informação geográfica na Web. Como tal, em 2.2.1 são descritas as duas formas de representar informação geográfica e em 2.2.2 os dois *standards* para partilha desta informação.

2.2.1 Modelos de Informação Geográfica

A informação geográfica consiste em dados que são referentes a um determinado lugar na Terra, tipicamente identificados por algum sistema de coordenadas. Para modelar este tipo de dados existem duas grandes aproximações: modelos vetoriais e modelos raster [51].

2.2.1.1 Modelos Vetoriais

Os modelos vetoriais usam vários tipos de dados para representar a realidade: pontos, linhas, polígonos, multipolígonos, entre outros. Os pontos são objetos que contêm um único par de coordenadas, tendo como única propriedade a sua localização. As linhas são o resultado de pontos explicitamente ligados entre si, possuindo como propriedade o seu comprimento. Por fim os polígonos são o resultado de múltiplas linhas ligadas entre si e que criam um espaço fechado, detendo como propriedades o seu perímetro e área.

Estes modelos são especialmente indicados para guardar dados que possuem fronteiras discretas como por exemplo parcelas agrícolas.

2.2.1.2 Modelos Raster

Os modelos raster representam o mundo a partir de uma matriz de células de igual tamanho. Essas células podem depois ser usadas para criar pontos, linhas e polígonos. Tipicamente cada célula terá associada a si os mais diversos atributos que forem de interesse bem como os valores correspondentes.

A precisão deste modelo varia consoante a resolução espacial (tamanho das células) que utiliza. Quanto maior for a resolução espacial, menor será a precisão.

Os modelos raster são bastante úteis para guardar dados que variam continuamente como por exemplo dados de satélite.

Na figura 2.2 podemos ver a comparação entre modelos vetoriais e modelos raster.

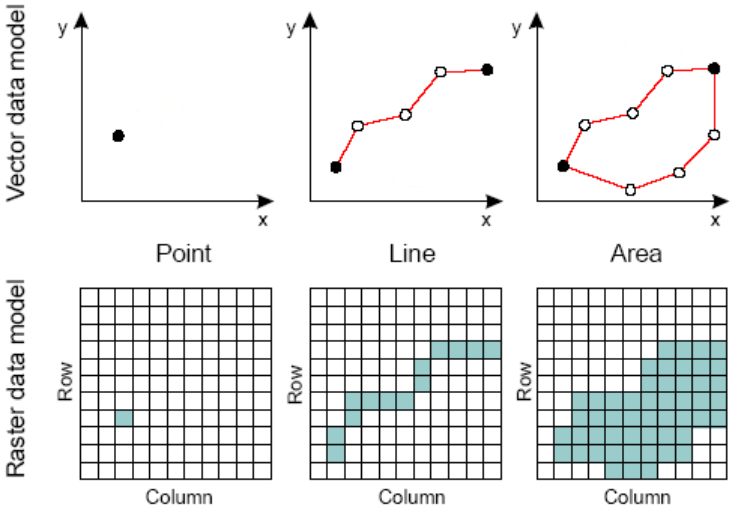


Figura 2.2: Comparação entre vetorial e raster [51]

2.2.2 Serviços de Informação Geográfica

O *Open Geospatial Consortium* (OGC) é uma organização internacional criada com a visão de um mundo em que todos beneficiam do uso de informação geográfica e de tecnologias relacionadas [26]. Para tal, a OGC dedica-se ao desenvolvimento de *standards*, aceites em larga escala pela comunidade, que se têm mostrado úteis para normalizar e melhorar a maneira como a informação geográfica é partilhada pela Internet.

De seguida serão apresentados de forma breve os dois *standards* estudados.

2.2.2.1 Web Map Service

O *Web Map Service* (WMS) [27] é um *standard* da OGC que define uma interface HTTP para pedidos de informação geográfica. Serviços deste tipo produzem dinamicamente mapas de informação georreferenciada a partir de informação geográfica. Este *standard* define "mapa" como sendo um retrato de informação geográfica, pronto a ser exibido num ecrã de um computador. Os mapas não são a informação geográfica em si mas sim uma

representação desses dados, normalmente renderizados em formato de imagem como PNG, GIF ou JPEG, ou eventualmente como elementos gráficos vectoriais em formatos como SVG ou WebCGM. O WMS classifica as informações geográficas em camadas e oferece um número finito de estilos nos quais pode exibir essas camadas.

Três operações são definidas para este serviço:

- **GetCapabilities** Operação obrigatória que devolve a metadata do serviço, ou seja, a descrição do conteúdo do servidor e os valores aceitáveis para os parâmetros de cada pedido. A resposta a estes pedidos é um documento XML contendo a metadata.
- **GetMap** Operação obrigatória que devolve um mapa. A resposta no caso do pedido ser válido é um mapa da camada de informação georreferenciada pedida no estilo desejado, seguindo todos os detalhes passados nos parâmetros (sistema de coordenadas, *bounding box*, tamanho, formato e transparência). No caso de ser um pedido inválido, uma exceção é lançada.
- **GetFeatureInfo** Operação opcional, apenas suportada pelas camadas que aceitem *queries*. É uma operação desenhada para proporcionar aos clientes de um serviço WMS mais informação sobre mapas devolvidos *a priori*.

2.2.2.2 Web Feature Service

O *Web Feature Service* (WFS) [28] é o outro *standard* definido pela OGC. Este *standard* define uma interface HTTP para descrever operações de manipulação de informação geográfica. Em vez de partilhar informação geográfica ao nível do ficheiro, o WFS oferece acesso direto aos dados ao nível de *features* e propriedades de *features*, permitindo que os clientes consultem ou modifiquem apenas a informação que procuram em vez de proporcionar acesso a um ficheiro com possivelmente informação a mais. Um pedido WFS consiste numa descrição da consulta e das operações de transformação a serem aplicadas aos dados. Após o pedido ser processado pelo servidor, a resposta vem normalmente em formato GML (*Geographic Markup Language*), uma gramática XML.

Este *standard* define onze operações:

- **GetCapabilities** Operação que devolve a metadata do serviço, descrevendo o serviço WFS providenciado pelo servidor.
- **DescribeFeatureType** Operação que retorna uma descrição dos tipos de *features* oferecidos pelo WFS.
- **GetPropertyValue** Operação que permite a obtenção do valor de uma propriedade de uma feature, a partir de uma *query*.
- **GetFeature** Operação que retorna um conjunto de *features*. Devolve ao cliente um documento que contém zero ou mais *features* que satisfazem a *query* especificada no pedido.

- **GetFeatureWithLock** Operação semelhante à GetFeature, só que para além de retornar o set de *features*, o WFS também dá *lock* às mesmas presumivelmente para ser feito um *update* numa operação *Transaction* futura.
- **LockFeature** Operação que dá *lock* a uma *feature*. Permite o controlo ao acesso de *features*, garantindo a consistência dos dados.
- **Transaction** Operação usada para descrever operações de manipulação de dados a serem aplicados a *features* de um WFS. Usando esta operação os clientes podem criar, modificar, substituir e eliminar *features*.
- **CreateStoredQuery** Operação que permite guardar uma *query* no servidor.
- **DropStoredQuery** Operação que permite a remoção de *queries* guardadas no servidor.
- **ListStoredQueries** Operação que lista as *queries* guardadas no servidor.
- **DescribeStoredQueries** Operação que disponibiliza metadata detalhada sobre cada *query* guardada no servidor.

2.3 Dados Meteorológicos

Os dados meteorológicos são parte nuclear desta dissertação. Como tal, nesta secção é apresentado o estudo que foi realizado no âmbito desta temática. Em 2.3.1 são apresentadas algumas normas de representação para este tipo de dados, em 2.3.2 são expostos alguns fornecedores de previsões e em 2.3.4 é referida uma ferramenta que interage com estações meteorológicas.

2.3.1 Normas Meteorológicas

Os dados meteorológicos apresentam desafios nomeadamente no que toca à sua representação, partilha e integração. Como tal, diversas comunidades e organizações têm vindo a propôr normas para este tipo de dados. Não havendo um único standard, foi feita uma pesquisa dos diversos formatos que representam dados meteorológicos.

De seguida, quatro desses formatos são descritos: BUFR, GRIB, NetCDF e HDF5.

2.3.1.1 BUFR

Binary Universal Form for the Representation of meteorological data (BUFR) [46, 62] é um formato de dados binário auto-descritivo, criado e mantido pela Organização Meteorológica Mundial (OMM), projetado para representar qualquer conjunto de dados meteorológicos que se sirva de um *stream* binário contínuo. O BUFR foi desenhado para conseguir armazenar e transferir informação meteorológica de forma eficiente.

Apresenta-se como um formato de dados *table-driven*, flexível e capaz de lidar com grandes volumes de informação.

Cada mensagem BUFR consiste num *stream* binário contínuo (2.3) composto por 6 secções, descritas na tabela 2.1.

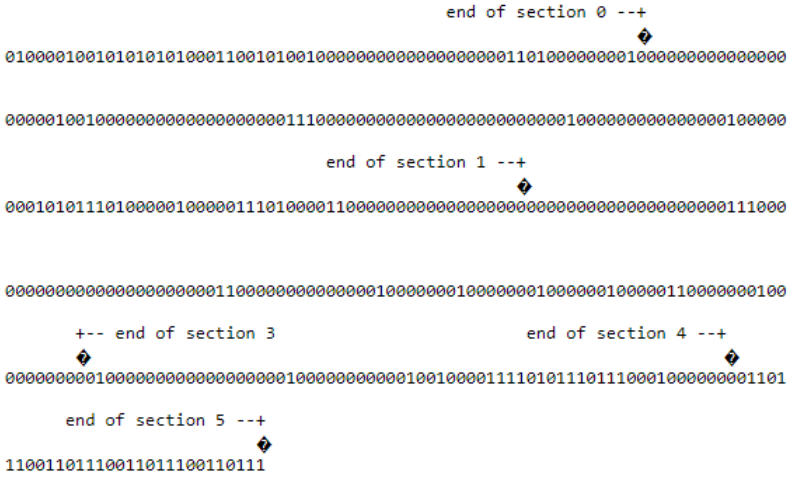


Figura 2.3: Exemplo de mensagem BUFR [46]

Secção	**Nome**	**Conteúdo**
0	Secção indicadora	Indica o início da mensagem usando a string "BUFR", o comprimento da mensagem e o número da edição BUFR
1	Secção de identificação	Comprimento da secção e descrição do conteúdo da mensagem
2	Secção opcional	Comprimento da secção e informação a ser utilizada pelos centros ADP (*Automated Data Processing*)
3	Secção de descrição dos dados	Comprimento da secção, número de *subsets* dos dados, *flag* de compressão, indicação se é uma observação ou uma previsão e lista de descritores que indicam a forma e o conteúdo dos dados
4	Secção dos dados	Comprimento da secção e os dados
5	Secção final	Indica o fim da mensagem com a *string* "7777"

Tabela 2.1: Composição de uma mensagem BUFR

2.3.1.2 GRIB

GRIdded Binary (GRIB) [47] é outro formato de dados binário auto-descritivo *table-driven* desenvolvido pela OMM, orientado ao armazenamento e transferência de dados meteorológicos. Tem duas edições (GRIB1 e GRIB2) sendo que a segunda edição foi desenvolvida para colmatar algumas deficiências da primeira. GRIB2 tem como principais características a flexibilidade e a expansibilidade, apresentando-se como uma forma eficiente de transmissão de grandes quantidades de dados em grelha.

Secção	Nome	Conteúdo
0	Secção indicadora	Indica o início da mensagem usando a *string* "GRIB", o comprimento da mensagem e o número da edição GRIB
1	Secção de identificação	Comprimento da secção e descrição do conteúdo da mensagem
2	Secção de uso local	Comprimento da secção e itens adicionais para uso local
3	Secção de definição da grelha	Comprimento da secção, definição da superfície da grelha e geometria dos dados dentro da superfície
4	Secção de definição do produto	Comprimento da secção e descrição da natureza dos dados
5	Secção de representação dos dados	Comprimento da secção e descrição de como os dados são representados
6	Secção do mapa de bits	Comprimento da secção e indicação da presença ou falta de dados para cada posição da grelha
7	Secção dos dados	Comprimento da secção e dados
8	Secção final	Indica o fim da mensagem com a *string* "7777"

Tabela 2.2: Composição de uma mensagem GRIB

Cada mensagem GRIB2 é composta por 9 secções, explicitadas na tabela 2.2.

Por serem binários, tanto o formato GRIB como o formato BUFR permitem que os dados sejam representados de forma mais compacta quando comparados com formatos orientados a caracteres, o que resulta em transmissões de informação mais rápidas.

Atualmente, *European Centre for Medium-Range Weather Forecasts* (ECMWF) possui a biblioteca ecCodes [13] que disponibiliza uma interface e um conjunto de ferramentas para codificar e descodificar mensagens GRIB1, GRIB2 e BUFR. Existe para C, Fortran 90 e Python.

2.3.1.3 NetCDF

Network Common Data Form (NetCDF) [52] é uma abstração de dados para armazenar e partilhar dados multidimensionais que sejam *array-oriented*. É distribuído como uma biblioteca que disponibiliza uma implementação concreta dessa abstração para Java, C, C++, Fortran [40] e Python [41]. Essa implementação providencia um formato *machine-independent* para representar dados científicos.

O NetCDF modela *data sets* científicos como um conjunto de variáveis multidimensionais, juntamente com o seu sistema de coordenadas e as suas propriedades.

Cada ficheiro NetCDF possui 3 componentes: dimensões, variáveis e atributos. Juntos permitem capturar o significado do *data set* científico que se está a representar.

Para representar dados meteorológicos, informações espaciais (latitude, longitude, nível

atmosférico) e o tempo são armazenadas como dimensões. As métricas (temperatura, pressão, humidade relativa, etc.) são armazenadas como variáveis. As variáveis são *arrays* de N-dimensões, com um nome e um conjunto de dimensões associadas. Os meta-dados de cada variável (unidades, intervalo válido, etc.) são armazenados como atributos.

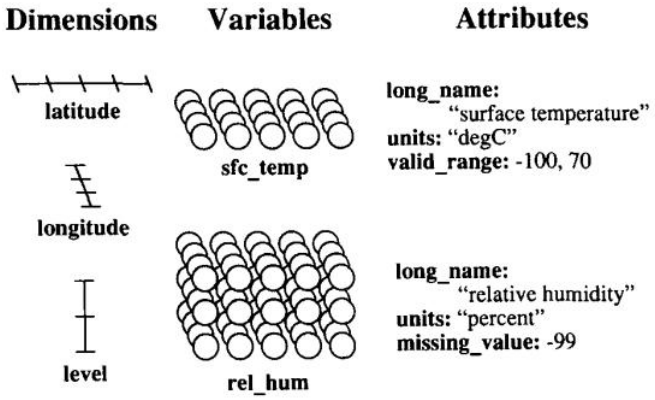


Figura 2.4: Exemplo de uma abstração NetCDF [52]

A figura 2.4 exemplifica uma abstração NetCDF para dados meteorológicos. Neste exemplo são consideradas como dimensões a latitude, a longitude e o nível atmosférico. As variáveis são a temperatura à surperfície, um *array* de duas dimensões (latitude e longitude), e a humidade relativa, um *array* de três dimensões (latitude, longitude e nível atmosférico). Por fim, cada variável possui os seus atributos. Esses atributos representam os meta-dados de cada variável.

2.3.1.4 HDF5

Hierarchical Data Format 5 (HDF5) [21] é um formato de dados flexível que se define pela forma hierárquica como organiza a informação. Foi desenhado para atender às exigências cada vez maiores no que toca à computação científica.

O HDF5 implementa um modelo para gerir e armazenar dados científicos. Esse modelo inclui um modelo abstrato de dados, um modelo abstrato de armazenamento (o formato) e ainda uma biblioteca que disponibiliza uma interface para a implementação dos modelos referidos previamente.

Ficheiros HDF5 são organizados numa estrutura hierárquica, constituída por:

- **Grupos**: estrutura que contém instâncias de zero ou mais grupos ou *datasets*, bem como os seus meta-dados. Cada grupo é composto por duas partes: um *header* que indica o nome do grupo e a lista de atributos que lhe correspondem e uma tabela de símbolos que diz respeito à lista de objetos HDF5 que pertencem ao grupo;

- ***Datasets***: um *array* multidimensional de dados bem como a meta-informação que lhe é associada. Cada *dataset* é constituído por duas partes: um *header* que contém

tanto a informação necessária para interpretar o *array* como também os meta-dados que descrevem o *dataset* e um *array* com os dados.

Qualquer grupo ou *dataset* pode ter ainda uma lista de atributos associada. Um atributo HDF5 é uma estrutura HDF5 definida pelo utilizador que disponibiliza informação adicional sobre um determinado objeto HDF5.

O h5py [23] e o JHI5 [31] são dois pacotes, para Python e Java respectivamente, que oferecem uma interface para manipulação de ficheiros HDF5.

2.3.2 Previsão Meteorológica

Atualmente existem diversos fornecedores online de dados de previsão meteorológica. IPMA [29], Accuweather [2] e OpenWeatherMap [45] são exemplos de fornecedores deste tipo de dados.

O IPMA disponibiliza uma API cujo acesso é livre e que faculta dados como previsão meteorológica diária até 5 dias agregada por local, previsão meteorológica diária até 3 dias agregada por dia e dados de sismicidade dos últimos 30 dias.

O Accuweather oferece diversos pacotes de subscrição para o acesso à sua API. Todos esses pacotes são pagos, excetuando o pacote de *trial* que limita o acesso à API a 50 chamadas por dia. Oferece dados como condições meteorológicas atuais, previsão para as próximas 12 horas e previsão para os próximos 5 dias.

Por fim, o OpenWeatherMap de forma semelhante ao Accuweather também oferece pacotes de subscrição para o acesso à API. São igualmente todos pagos, excluindo o pacote *free* que está a limitado a 60 chamadas por minuto. Disponibiliza dados como condições meteorológicas atuais, previsão para as próximas 3 horas e previsão para os próximos 5 dias.

2.3.3 Convenções de Nomenclatura

Atualmente não existe nenhuma convenção de nomenclatura para dados meteorológicos que seja aceite e utilizada globalmente. Isto significa que diferentes fornecedores deste tipo de dados muitas vezes utilizam nomenclaturas diferentes para o mesmo tipo de dados.

Como tal, o METNorway (*Norwegian Meteorological Institute*) propôs um *standard* [3] para este tipo de dados com o objetivo primário dos identificadores utilizados para dados e observações meteorológicas serem:

- **Auto-descritivos** Os utilizadores devem ser capazes de ler e compreender os identificadores, reduzindo o risco de má utilização dos dados.
- **Consistentes** O mesmo identificador significará sempre o mesmo em todos os contextos, identificadores diferentes dirão sempre respeito a dados diferentes.
- **User-friendly** Fácil de utilizar por pessoas que não sejam profissionais em geociências.

A convenção proposta constrói identificadores segundo a regra <method>(<base name> <period>) sendo que permite *inner methods*. Cada componente desta regra é definido da seguinte maneira:

- **method** O método principal (função) aplicado diretamente à quantidade primária ou indiretamente via um *inner method*.
- **base name** A quantidade primária. Um nome *standard* de acordo com a CF *standard name table* [7] ou construído segundo as CF *guidelines* [22] para a construção de nomes *standard*.
- **period** O período sobre o qual o *method* é aplicado.

De seguida são apresentados dois exemplos deste *standard* para melhor compreensão:

- **mean(air_temperature PT1H)** Média horária da temperatura do ar.
- **mean(max(wind_speed P1D) P1M)** Média mensal do máximo diário da velocidade do vento.

2.3.4 weeWX

O programa weeWX [60] é um software *open source*, escrito em Python, que interage com estações meteorológicas para produzir gráficos, relatórios e até páginas HTML. Para além destas funcionalidades oferece ainda correções de calibração e filtragem de valores anómalos, apresentando-se como uma ferramenta robusta e com um carácter extensível, abrindo portas à implementação de novos serviços e relatórios. O modelo de dados utilizado por este software consiste numa tabela de arquivo em que cada registo contém todas as métricas medidas pela estação num dado ponto temporal.

Na figura 2.5 podemos ver a aplicação desta ferramenta a uma estação meteorológica situada em Royston, Reino Unido.

2.4 Interpolação Espacial

A informação gerada pelas estações meteorológicas é apenas referente à localização onde estas se encontram e a informação de satélite pode não estar disponível em certos intervalos temporais para certas regiões devido à presença de, por exemplo, nuvens que não permitem a recolha de imagens e consequentemente dados. Como tal, para se garantir que a informação é o mais densa possível, isto é, para que hajam dados disponíveis para qualquer região a qualquer altura temporal, torna-se necessário recorrer a métodos de interpolação espacial e temporal. Estes métodos permitem utilizar regiões com valores conhecidos para estimar os valores em falta de outras regiões.

Em [36] são discutidos diversos métodos de interpolação espacial, tendo em conta os mais diversos fatores (distribuição espacial da amostra, qualidade dos dados, performance,

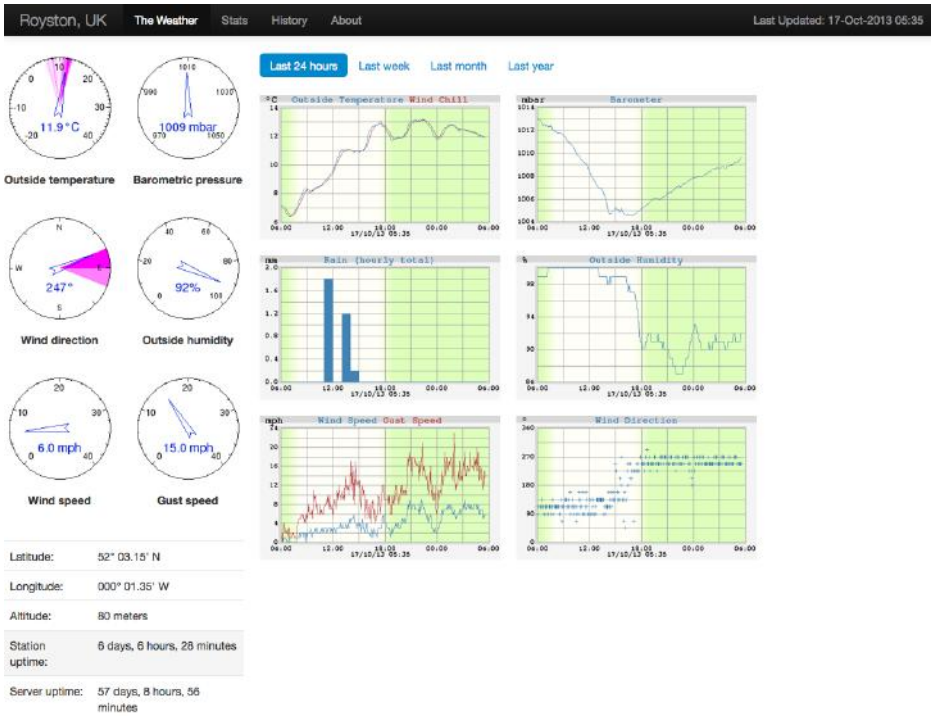


Figura 2.5: Exemplo de aplicação do programa weeWX [60]

entre outros) sugerindo em que situações é melhor utilizar cada um. De seguida serão descritos de forma breve três métodos de interpolação espacial.

2.4.1 Nearest neighbour

A interpolação *nearest neighbour* (vizinho mais próximo) é um método bastante simples que estima o valor desconhecido tendo meramente em consideração o valor conhecido mais próximo. O valor estimado irá corresponder exatamente ao valor mais próximo.

2.4.2 Inverse distance weighting

O método *inverse distance weighting* (IDW) é semelhante ao método *nearest neighbour* mas em vez de considerar unicamente o valor da região mais próxima, considera múltiplas regiões atribuindo pesos diferentes consoante a distância à região a ser determinada (quanto maior a distância, menor o peso). Para reforçar o peso das regiões mais próximas da região que se quer estimar, existe ainda a possibilidade de aplicar uma potência à distância entre estas.

$$z_p = \frac{\sum_{i-1}^{n} \frac{1}{d_i^p} z_i}{\sum_{i-1}^{n} \frac{1}{d_i^p}} \tag{2.1}$$

Em 2.1 podemos ver a equação que nos permite estimar valores a partir deste método, sendo que:

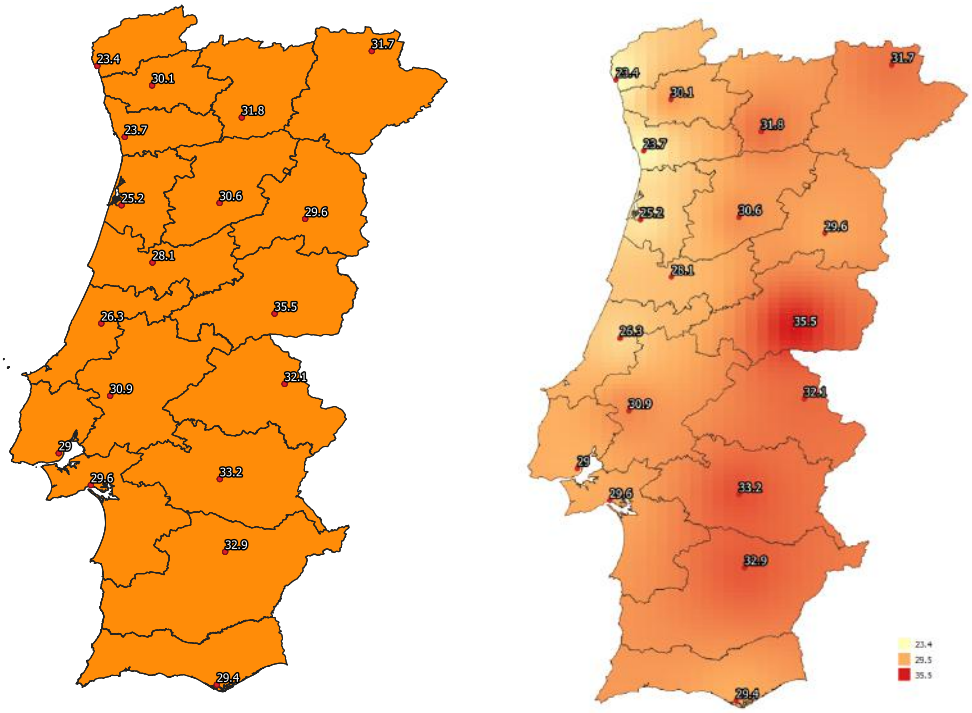
z_p = valor estimado considerando a potência p;

n = número de regiões consideradas;

z_i = valor da i-ésima região;

d_i^p = potência da distância entre a i-ésima região e a região a ser estimada.

Na figura 2.6 é apresentada uma aplicação deste método no software QGIS. Em 2.6a estão representados os dados da temperatura máxima referentes ao dia 10-07-2018, adquiridos através da API do IPMA. Em 2.6b vê-se o resultado da execução do método IDW.



(a) Dados IPMA (b) Resultado de IDW

Figura 2.6: Exemplo de aplicação de IDW

2.4.3 Kriging

Kriging é um método de interpolação geoestatístico. Abordagens geoestatísticas são usadas para descrever padrões espaciais e interpolar valores desconhecidos, modelando ao mesmo tempo a incerteza ou erro da estimativa realizada. De forma semelhante ao método IWD atribui pesos aos valores próximos conhecidos, pesos esses que têm em consideração não só a distância ao valor que se quer estimar mas também a distribuição espacial das amostras consideradas.

Existem diversas variantes deste método, todas referidas em [36]. Uma das variantes mais utilizadas é *Ordinary Kriging*, abordada de seguida.

$$\hat{Z}(s_0) = \sum_{i=1}^{n} \lambda_i Z(s_i) \tag{2.2}$$

Em 2.2 podemos ver a equação que nos permite estimar valores a partir do método *Ordinary Kriging*, em que:

$Z(s_i)$ = valor da i-ésima região;

λ_i = um peso desconhecido para o valor da i-ésima região;

s_0 = região que se quer estimar;

n = número de regiões consideradas.

2.5 Dados de Satélite

A informação de satélite é obtida a partir de imagens de satélite onde para cada pixel da imagem é possível obter a latitude, a longitude e o valor do parâmetro que se estiver a considerar.

Esta secção apresenta o estudo dos dois tipos de dados de satélite que foram considerados para integração no sistema de informação a ser desenvolvido: dados *land surface temperature* e *dados land cover*. Para ambos é feita uma pequena descrição sobre o que são, da sua utilidade e por fim, das fontes que disponibilizam cada um deste tipo de dados.

2.5.1 Dados Land Surface Temperature

Os dados LST correspondem à temperatura ao solo terrestre calculada a partir da radiação emitida pela Terra. Em [55] foi realizado um estudo comparativo entre o uso de dados de estações meteorológicas e o uso de dados LST na execução de modelos de previsão de pragas para geração de mapas de risco que revelou uma relação linear entre ambos.

Como tal, os dados LST são bastante importantes para complementar a informação proveniente de estações. Enquanto que os dados provenientes de estações meteorológicas dizem respeito exclusivamente à localização onde estas se encontram, os dados LST por serem recolhidos via satélite permitem fazer uma cobertura global da superfície terrestre.

De seguida será abordada uma fonte de dados LST - EUMETSAT.

2.5.1.1 EUMETSAT

Os satélites da *European Organization for Exploitation of Meteorological Satellites* (EUMETSAT) [15] recolhem dados meteorológicos e climáticos como *land surface temperature*, evapotranspiração, cobertura vegetal, entre outros. Essa informação é disponibilizada livremente em ficheiros HDF5 (2.3.1.4) por um serviço FTP do sistema *Satellite Applications Facility on Land Surface Analysis* (LSA SAF) [37], sistema que conta com a participação do IPMA.

Falando especificamente dos dados LST [33], a informação disponibilizada pelo LSA SAF possui uma resolução espacial de aproximadamente 3 por 3 kms nas coordenadas

do território português e uma resolução temporal de 15 em 15 minutos, o que representa 96 medições diárias para cada um dos quase 7000 locais. Na figura 2.7 podemos observar dados LST disponibilizados pelo LSA SAF do dia 28 de Junho de 2018 às 18 horas. As diferentes cores representam a temperatura ao solo terrestre consoante os valores impostos na escala presente à direita na figura.

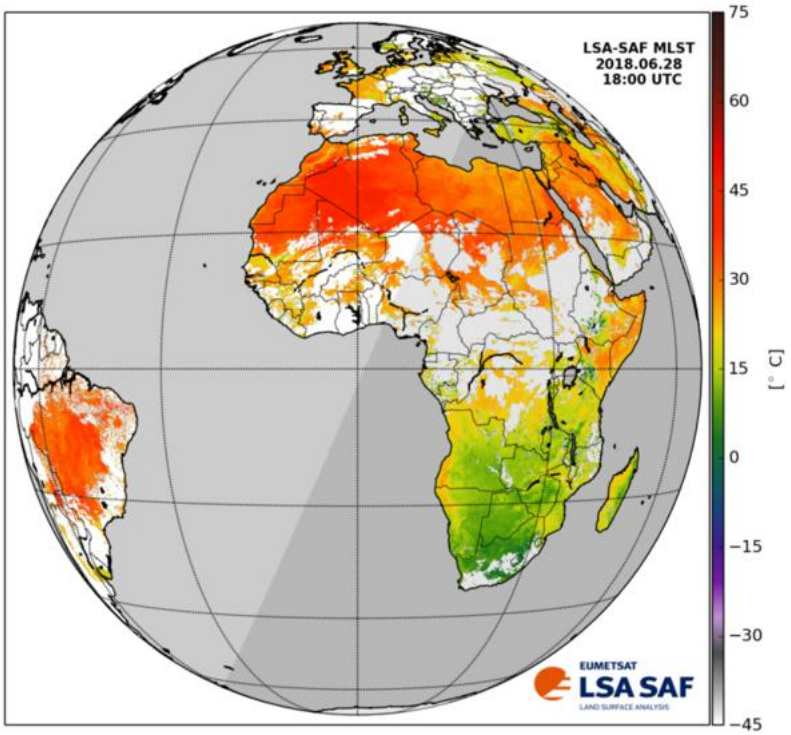


Figura 2.7: Exemplo de dados LST recolhidos por satélite [37]

A figura 2.7 expõe um dos problemas de dados provenientes de satélites. A existência de nuvens numa determinada região (áreas brancas na figura 2.7) impossibilita a recolha de informação. Esse problema motivou o estudo de métodos de interpolação espacial e temporal (2.4) que permitem estimar os dados em falta.

2.5.2 Dados Land Cover

Os dados *land cover* provenientes de satélite permitem mapear os diferentes tipos de cobertura terrestre o que contribui para uma melhor caracterização da superfície da Terra. No contexto deste projeto, possuir este tipo de dados de forma contínua ao longo do tempo ajuda a monitorizar e a detetar alterações na vegetação dos campos agrícolas. Foram feitas pesquisas sobre dois satélites que disponibilizam o acesso a este tipo de dados de forma livre: Landsat-8 e Sentinel-2.

2.5.2.1 Landsat-8

O satélite Landsat-8 [30], desenvolvido pela *National Aeronautics and Space Administration* (NASA) e pela *United States Geological Survey* (USGS), recolhe imagens da superfície terrestre a oito bandas espetrais diferentes, com resolução espacial de 30 metros e temporal de 16 dias. Toda a informação recolhida por este satélite está disponível por via de uma API mantida pela USGS. Em 2.8 podemos ver uma imagem recolhida pelo Landsat-8.

Figura 2.8: Imagem recolhida pelo satélite Landsat8 [58]

2.5.2.2 Sentinel-2

O satélite Sentinel-2 [12], lançado pela *European Space Agency* (ESA), recolhe imagens a treze bandas espetrais diferentes, com uma resolução espacial que varia entre os 10, 20 e 60 metros (consoante a banda espetral em questão) em ciclos de 5 dias.

Tem como objetivo principal a recolha sistemática de imagens multi-espetrais de alta resolução da superfície terrestre, contribuindo para a geração de produtos como *land cover maps*, *land change detection maps* e variáveis geofísicas. Todos os dados são acessíveis a partir de um sistema mantido pela ESA, requerendo apenas um registo prévio.

Em [25] foi feito um estudo que incidiu no uso dos dados deste satélite para mapeamento de culturas e espécies de árvores em campos agrícolas situados na Europa Central. Este estudo concluiu que os resultados obtidos foram satisfatórios e que podem ser melhorados caso a ESA melhore a qualidade dos dados ou caso o *timing* de recolha seja ótimo.

2.6 Tecnologias Relevantes

Tanto os dados recolhidos em campo como os dados meteorológicos (das mais diversas fontes) e os dados de satélite apresentam um carácter espácio-temporal. Devido a este facto, é apresentado nesta secção o estudo do estado da arte em tecnologias que acomodam dados com este tipo de características. Para além de bases de dados relacionais, foram ainda

consideradas tecnologias Big Data devido ao problema espacial que os dados de satélite apresentam. Por fim, são também apresentadas as tecnologias que foram consideradas para o desenvolvimento do servidor deste sistema. Foram consideradas tecnologias estáveis e testadas, largamente aceites e utilizadas pela comunidade, devido à necessidade de futura manutenção e extensão de funcionalidades.

2.6.1 PostgreSQL

PostgreSQL [50] é um sistema de gestão de base de dados *open source* que apresenta uma arquitetura confiável, tendo em conta a integridade, a consistência e a replicação dos dados. Este sistema é nos dias de hoje um dos sistemas de base de dados relacional mais avançados já que possui recursos sofisticados que utilizam por exemplo o método de controlo multi-versão para lidar com a consistência dos dados em acessos concorrentes à mesma tabela, ou o método de replicação lógica que permite um controlo refinado sobre a replicação e a segurança dos dados. Apresenta-se como sendo um sistema robusto, capaz de responder a mecanismos complexos e de lidar com bases de dados de grandes dimensões.

Possui a extensão PostGIS [49] o que lhe confere capacidades de representação de informação geográfica já que adiciona tipos de pontos geométricos, multipontos, retas, polígonos e multipolígonos. Esta extensão permite que o *PostgreSQL* seja utilizado enquanto base de dados para sistemas de informação geográfica.

2.6.2 SQLite

SQLite [57] é uma biblioteca em C que implementa uma base de dados transacional *stand-alone* embebida. Ao contrário da maior parte das bases de dados SQL, SQLite não tem um processo servidor em separado. Programas que usem esta biblioteca lêem e escrevem diretamente nos ficheiros de base de dados no disco. É uma base de dados com reputação de ser bastante confiável, lidando bem com falhas de alocação de memória e erros de *input/output* no disco. Garante ainda transações ACID mesmo no caso do sistema *crashar* ou de haver alguma falha de energia. No entanto não lida bem com grandes quantidades de dados.

SpatiaLite [56] é uma biblioteca *open source* desenvolvida para ser integrada com SQLite, oferecendo suporte a capacidades de representação de informação geográfica (de forma semelhante ao que o PostGIS oferece ao PostgreSQL).

2.6.3 Secondo

Secondo [42] é uma base de dados extensível e escalável, escrita em C++, Prolog e Java, desenvolvida pela FernUniversität em Hagen. Fornece suporte para os mais diversos modelos de dados (e.g. relacional, espacial, espaço-temporal) e funções (e.g. *filter*, *aggregates*, *joins* e *spatial-joins*), permitindo *queries* espaço-temporais.

Combinado com Apache Cassandra, oferece uma base de dados distribuída, altamente escalável e disponível, capaz de acomodar taxas de *input* elevadas e *queries* complexas.

2.6.4 ST-Hadoop

ST-Hadoop [4] é uma *framework* MapReduce com o objetivo de disponibilizar suporte nativo para armazenamento e pesquisa de dados espaço-temporais. Este *framework* dá suporte à criação de um índice espaço-temporal, replicado a diferentes granularidades temporais, que permite melhorar o desempenho de pesquisas com diferentes intervalos de tempo.

Utiliza uma extensão da linguagem Pigeon [14] para as pesquisas, visto que esta adiciona suporte para tipos de dados temporais (STPoint, TIME, INTERVAL) e também para diferentes predicados (OVERLAP, JOIN). A figura 2.9 apresenta um exemplo de pesquisa espaço-temporal usando esta linguagem.

```
Objects = LOAD 'points' AS (id:int, STPoint:(Location,Time));
Result  = FILTER Objects BY
          Overlaps (STPoint, Rectangle(x1, y1, x2, y2), Interval (t1, t2) );
```

Figura 2.9: Pesquisa por intervalo de tempo e espaço usando Pigeon [4]

Dois tipos de pesquisa espaço-temporal são suportados:

- **ST-Range Query** Pesquisa especificada por dois predicados, uma área espacial (A) e um intervalo temporal (T). Retorna o conjunto de elementos que se encontrem na região A e no intervalo de tempo T.
- **ST Join** Dados dois *datasets* (r e s) contendo elementos espaço-temporais indexados, e um predicado P, esta pesquisa retorna todos os pares de elementos <r,s> que satisfaçam o predicado P.

2.6.5 Geomesa

Geomesa [24] é uma base de dados espaço-temporal distribuída, que se apoia na estrutura de armazenamento de dados KV (Key-Value) da Accumulo [1]. Inclui ferramentas que possibilitam a indexação, a gestão e a análise de dados raster e dados vetoriais. Para indexar dados, tendo em conta a sua componente espacial e temporal, são utilizadas *space filling curves* que permitem o mapeamento de dados multidimensionais para uma dimensão apenas, mantendo a localidade dos dados. Na figura 2.10 é apresentado um exemplo de uma *space filling curve*.

Os valores são armazenados na base de dados utilizando uma chave que contém a informação espaço-temporal do objeto a ser introduzido.

Suporta a linguagem ECQL (*Extended Common Query Language*) que é uma extensão da linguagem CQL (*Common Query Language*) definida pelo *Open Geospatial Consortion*

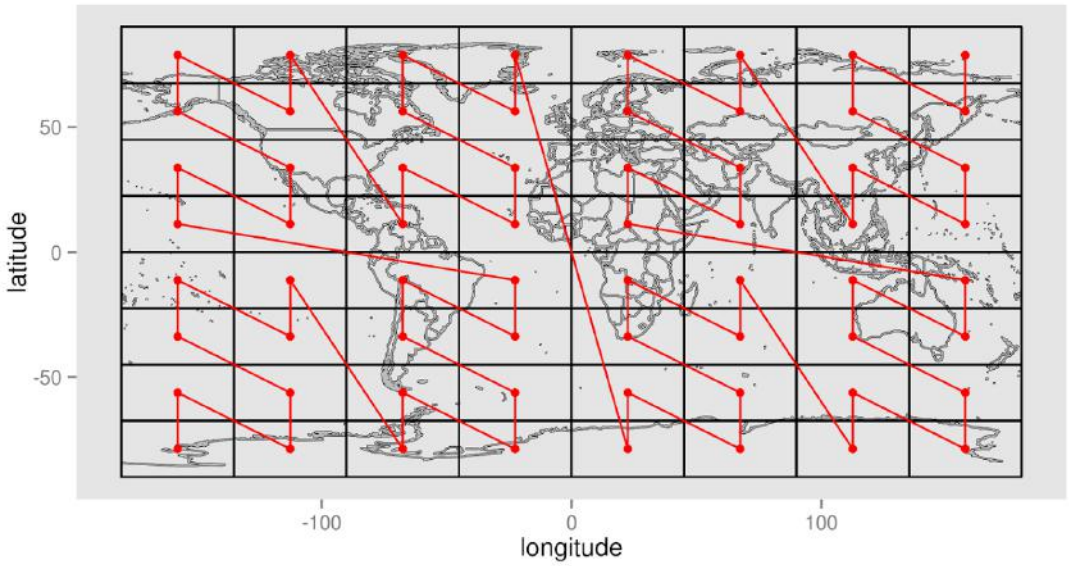


Figura 2.10: Exemplo de uma space filling curve (Z-order curve) [20]

(referido em 2.2). Esta linguagem oferece operadores espaciais para suportar pesquisas contendo intervalos ou interseções espaciais e também operadores temporais como BEFORE, AFTER e DURING.

2.6.6 Flask

Flask [19] é um *microframework* Python destinado principalmente ao desenvolvimento de pequenas aplicações Web com requisitos simples. Desenhado para ser fácil de usar e estender, tem ainda inúmeras extensões e *plug-ins* ao seu dispor. Possui um *development server* e *debugger built-in*, suporta *web-services* REST e tem suporte nativo para testes unitários.

Dentro dos *microframeworks* disponíveis atualmente, Flask é dos mais polidos. Tem uma comunidade ativa e está extensivamente documentado.

2.6.7 Django

Django [11] é um *framework* Python para o desenvolvimento de aplicações Web que utiliza o padrão *Model-Template-View* (MTV). Concebido para a construção rápida de software, conta com mais de dez mil *packages* com as mais diversas funcionalidades.

Uma das grandes vantagens do *framework* Django é possuir um *object-relational mapper* integrado, o que facilita a interação com as bases de dados.

Escalável, seguro e bem documentado, Django é utilizado por empresas como NASA, Instagram, Reddit e The Washington Post.

2.6.8 Ruby on Rails

Ruby on Rails [54] é um *framework* para desenvolvimento de aplicações baseada na linguagem de programação Ruby. Apresenta-se como um *framework* completo que segue a arquitetura *Model-View-Controller* (MVC), promovendo assim a modularidade e a extensibilidade. Devido à enorme quantidade de bibliotecas *open source* para os mais diversos propósitos, Ruby on Rails é ótimo para desenvolvimento rápido de software. Porém, denota pior performance e velocidade de execução quando comparado com outros *frameworks* (nomeadamente Django e Node.js), o que pode comprometer a escalabilidade do sistema.

2.6.9 Node.js

Node.js [43] é um ambiente de execução Javascript com uma arquitetura *event-driven*, capaz de tratar *input/output* assíncrono. Estas escolhas arquiteturais procuram otimizar o rendimento e a escalabilidade de aplicações Web que lidam com um elevado número de operações de *input* e *output*.

Tem ao seu dispor uma biblioteca com mais de seiscentos mil módulos que facilitam o desenvolvimento Web. Todos os módulos são *open-source* e podem ser melhorados, corrigidos ou adaptados por qualquer programador.

2.7 Conclusões

Este capítulo resultou no estudo e na aprendizagem de diversas temáticas consideradas relevantes para esta dissertação.

Os trabalhos relacionados permitiram perceber que soluções foram encontradas para problemas semelhantes. O estudo de dados meteorológicos resultou na aprendizagem das diversas formas de os representar. Os dados de satélite (LST e *land cover*) mostraram-se como dados importantes no âmbito do problema que se pretende resolver, com a vantagem de terem livre acesso. As técnicas de interpolação espacial revelaram-se um valioso recurso para se terem dados contínuos tanto espacial como temporalmente. Todo o estudo realizado no âmbito da informação geográfica permitiu a aprendizagem de como a mesma se representa e dos *standards* para a sua partilha na Internet (WMS e WFS).

Por fim, toda a informação recolhida relativamente às tecnologias espaço-temporais e às tecnologias servidor permitirá uma escolha racional e fundamentada no que toca às tecnologias e utilizar na implementação do sistema de informação FitoAgro.

Capítulo 3

Modelação

O presente capítulo tem como objetivo apresentar os resultados finais de todo o processo de modelação da solução que foi implementada.

Numa primeira fase é apresentada uma introdução ao sistema de informação, referindo os seus conceitos e requisitos principais. Posteriormente é discutida a arquitetura adotada, o modelo de dados desenhado, a modelação da componente de data warehousing, a API REST e, por fim, a metodologia de desenvolvimento utilizada.

3.1 Introdução

A presente dissertação, tal como foi referido em 1.4 tem como objetivo a implementação de um sistema de informação de base que permita centralizar todos os dados gerados por projetos de investigação da área agrícola, focados no estudo, prevenção e controlo de pragas - no âmbito desta dissertação será aplicado ao projeto FitoAgro.

No passado foram realizadas implementações de sistemas de informação para este domínio, nomeadamente no âmbito do projeto Protomate e do projeto SafeBrocolo, no entanto o resultado final foi uma implementação mais específica que não possibilitou a sua reutilização para projetos subsequentes.

Este tipo de projetos geram principalmente dois tipos de dados: dados de campo e dados meteorológicos. Para que se possa perceber melhor, irei concretizar com o caso do projeto FitoAgro. Neste projeto estão então a ser gerados:

- Dados de campo: Técnicos deslocam-se periodicamente ao campo, onde registam em uma ou mais folhas de registo o que observam (consoante protocolos previamente definidos). Posteriormente esses dados são passados para folhas de cálculo Google Sheets que estão numa Drive do projeto.

- Dados meteorológicos: Parceiros do projeto adquiriram múltiplas estações meteorológicas, nem sempre do mesmo fornecedor. Isto implica que os dados tenham diferentes formas de acesso e diferentes formas de representação.

Tendo em conta tudo o que foi referido, os requisitos para o sistema de informação são os seguintes:

- Informação do Projeto: O sistema deverá ter a capacidade de obter e armazenar toda a informação gerada pelo projeto.
- Facilidade na disponibilização da informação: O sistema deverá ser capaz de disponibilizar toda a informação para consulta e manipulação através de *web-services* fáceis de utilizar.
- Função Analítica: O sistema deverá dar resposta às necessidades analíticas do projeto.
- Genérico: O sistema desenvolvido deverá ser tão genérico quanto possível, de forma a que possa ser aplicado a projetos do mesmo domínio sem que com a genericidade se torne difícil de utilizar.
- Extensível: O sistema desenvolvido deverá ter a capacidade de ser estendido no caso do aparecimento de novos requisitos ou novas fontes de dados.
- Auditável: A todo o momento o sistema deverá ter a capacidade de poder ser auditável devido a receber dados de múltiplas fontes. Dessa forma, todos os dados que entram no sistema devem poder ser rastreados.

3.2 Arquitetura

A arquitetura do sistema de informação segue um estilo cliente-servidor e encontrando-se o seu diagrama na figura 3.1. Neste diagrama podem ser observadas ver as quatro camadas principais da arquitetura: Camada Cliente, Camada Servidor, Camada de Armazenamento e Camada de Visualização.

A Camada Cliente não foi desenvolvida no âmbito desta dissertação, no entanto será a camada responsável por toda a interacção dos utilizadores com o sistema, servindo-se da API REST para consultar e/ou manipular a informação.

A Camada Servidor para além de disponibilizar a API REST, é também o local onde reside toda a lógica do negócio, responsável por processar os pedidos que chegam pela API, bem como o ORM (object-relational mapping) utilizado para comunicar com a base de dados. Adicionalmente, esta camada tem um componente denominado por Integrador de Dados que consiste num conjunto de *scripts* responsáveis por adquirir, tratar e integrar os dados provenientes das mais diversas fontes.

A Camada de Armazenamento corresponde à base de dados onde toda a informação do sistema fica persistida. A base de dados possui dois componentes distintos - por um

lado um modelo de dados para suportar as funcionalidades operacionais e por outro um modelo de dados orientado à analítica (Data Warehouse).

Por fim, a camada de visualização consiste num conjunto de visualizações realizadas através dos dados disponibilizados pelo Data Warehouse que permitem responder às necessidades analíticas existentes.

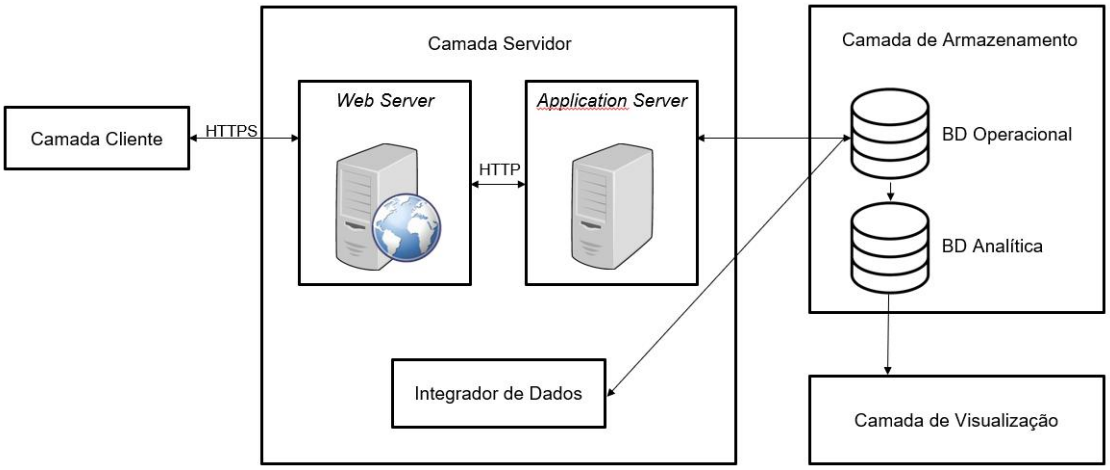


Figura 3.1: Arquitetura do sistema

3.3 Modelo de Dados

A modelação do modelo de dados foi um processo moroso e complexo que requereu a tomada de um certo nível de abstração visto se estar a modelar o domínio de projetos de monitorização fitossanitária e não apenas o projeto FitoAgro.

Para simplificar a explicação do modelo de dados do sistema, numa primeira fase serão apresentadas as regras de desenho 3.3.1 definidas e seguidas ao longo de toda a modelação. Após este ponto introdutório e para facilitar a compreensão, o modelo será apresentado por cada um dos seus módulos identificados por cores: Espaço-Temporal 3.3.2 (módulo azul), Agro-Negócio 3.3.3 (módulo laranja), Dados de Campo 3.3.4 (módulo verde), Séries Temporais 3.3.5 (módulo vermelho), Agentes e Organizações 3.3.6 (módulo amarelo) e Configurações 3.3.7 (módulo cinzento). Para cada um destes módulos, além do diagrama, serão apresentadas e discutidas cada uma das suas entidades consideradas principais. A discussão irá cingir-se às entidades principais para que o foco não se perca durante a explicação de um modelo de dados que é complexo.

De modo a facilitar a leitura do documento, cada uma das entidades, apesar de ter o seu nome em inglês nos diagramas, será mapeada para um nome correspondente em português que será utilizado ao longo do documento. Todo o mapeamento de nomes poderá ser consultado no apêndice A. Na figura 3.2, de modo a ser conseguida uma panorâmica do modelo concebido, é possível ver um diagrama que mostra apenas as entidades e as relações existentes entre si.

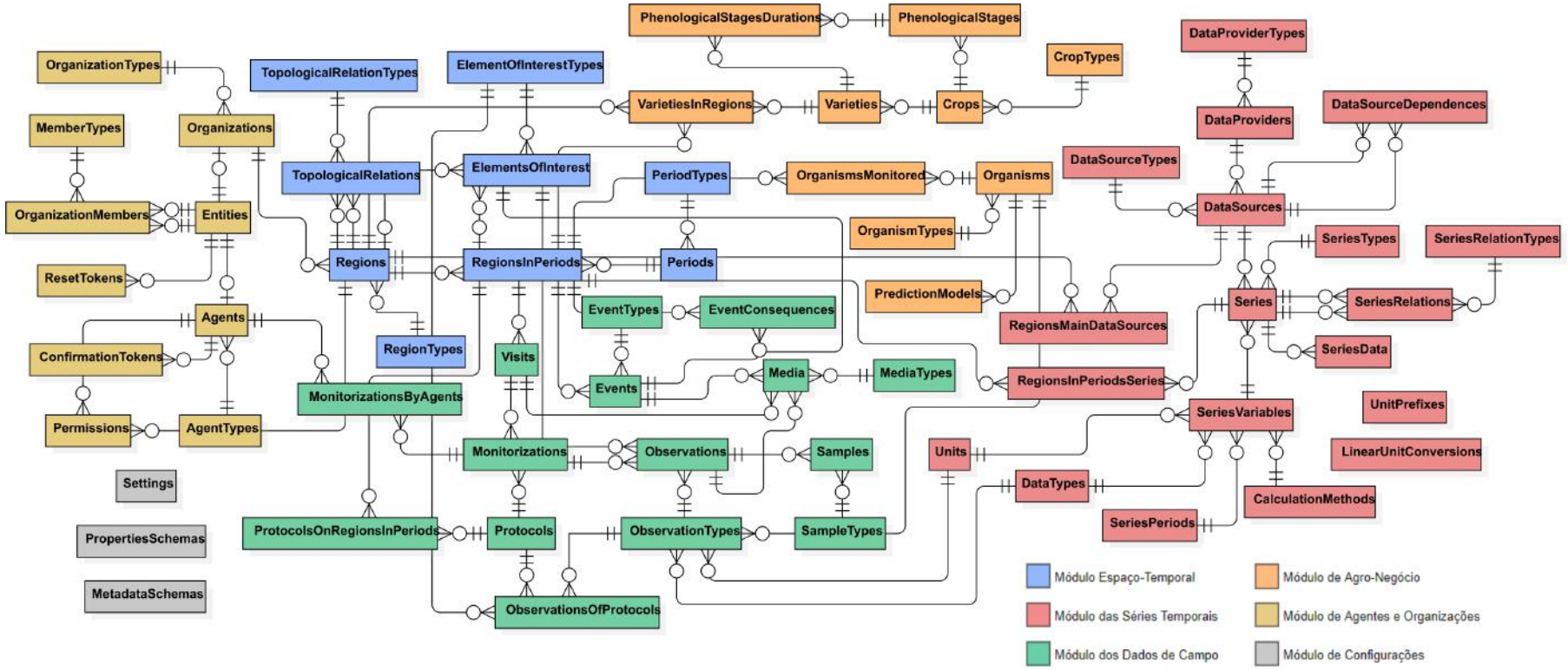


Figura 3.2: Vista geral do modelo de dados

3.3.1 Regras de Desenho

Ao longo da conceção do modelo de dados, foi definido um conjunto de regras de desenho que funcionaram como fundamentos a seguir ao longo de todo o processo de modelação. Como tal, serão de seguida apresentadas cada uma dessas regras, acompanhadas de uma breve explicação do porquê de terem sido estabelecidas e aplicadas, de forma a facilitar a compreensão do modelo que será apresentado em mais detalhe nas secções subsequentes.

- **Nomenclatura de Tabelas**

  Todas as tabelas do modelo têm o seu nome no plural devido a representarem um conjunto de objetos/entidades em questão. Para além disso, houve uma preocupação constante de adotar nomes tão genéricos quanto possível, isto é, que permitissem um nível de abstracção grande sem perda total de significado.

- **Nomenclatura de Atributos**

  Os atributos seguem a regra de nomenclatura <nome_tabela_singular>_<nome_atr ibuto>. Optou-se por utilizar o nome da tabela concatenado com o nome do atributo para evitar ambiguidades que poderiam ocorrer devido à existência atributos com nomes iguais em tabelas diferentes.

- **Atributo Properties**

  Foi criado um atributo **Properties** em todas as tabelas. Esse atributo é um objeto JSON que surgiu para dar elasticidade ao modelo, permitindo que a qualquer momento possam ser adicionados novos atributos à tabela sem a alterar, conferindo-lhe assim um carácter extensível. No entanto, a utilização de objetos JSON numa base de dados pode tornar-se caótica - devido à facilidade de criar um novo par *key-value* no objeto, facilmente se pode perder a noção da sua estrutura. Para evitar esse problema, o esquema desse objeto JSON é definido *a priori*, sendo que o atributo Properties pode ter: propriedades globais da entidade em questão, cujo esquema é definido numa tabela auxiliar criada somente para esse propósito e propriedades específicas de tipos (aplicável quando a entidade em questão tem um tipo associado) cujo o esquema estará definido num atributo da tabela que define o tipo.

- **Atributo Metadata**

  De forma similar ao atributo Properties, todas as tabelas têm um atributo JSON denominado **Metadata** com o objetivo de guardar metadados dos dados. Esse metadados poderão servir, dependendo do projeto em questão, para tornar o sistema de informação *semantic web aware*, para fazer controlo de acessos, entre outros. Para evitar o problema da utilização de atributos JSON (referido no ponto acima) foi também criada uma tabela auxiliar que define para cada tabela o esquema JSON a ser utilizado para o atributo Metadata. Com esta abordagem foi pretendido dar

completa liberdade para se decidir, consoante os requisitos de cada projeto, como utilizar o campo de metadados.

- **Estampilhas Temporais**

  Três estampilhas temporais estão presentes em todas as tabelas, sendo estas: **created_at**, **updated_at** e **deleted_at**. As duas primeiras indicam simplesmente quando o registo foi criado e atualizado respectivamente. A estampilha deleted_at foi criada segundo a filosofia de que nada no sistema de informação deve ser realmente removido (filosofia de soft-deletes). Em vez disso, no caso de haver algum *delete* de dados, o campo deleted_at é atualizado sem que o registo seja eliminado da base de dados. Desta forma é possível haver uma fácil recuperação de informação no caso de enganos por parte dos utilizadores.

- **Tabelas de Associação**

  As tabelas de associação (tabelas resultantes de relações N:M) foram desenhadas com uma *surrogate key* enquanto chave primária ao contrário de usar a combinação das chaves primárias das tabelas que estão em cada respectiva associação. O racional por detrás dessa decisão prende-se com o carácter extensível que se pretende atribuir ao modelo. Caso no futuro se queira referenciar uma determinada associação numa nova tabela, associação essa que devido à flexibilidade conferida pelo atributo Properties tem a capacidade de se tornar uma entidade em si, basta apenas recorrer à tal *surrogate key*.

- **Tabelas de Tipos**

  Sempre que aplicável, as tabelas do modelo têm relacionadas a si uma tabela de tipos. Estas tabelas de tipos contribuem diretamente para atingir o objetivo de tornar o modelo de dados genérico uma vez que atuam de certa forma como classificadores da tabela com que se relacionam, suportando a generalização do seu conteúdo.

3.3.2 Módulo Espaço-Temporal

Este módulo é identificado pela cor azul e procura modelar os aspetos espaço-temporais do modelo. Na figura 3.3 é possível observar o diagrama de tabelas deste módulo.

3.3.2.1 Períodos

Os períodos modelam intervalos temporais exatos, ou seja, com um data de início e com um data de fim definidas. Estes períodos, para além da sua data de início e fim, contam com informação como o seu nome e uma descrição opcional. Para o projeto FitoAgro, um dos períodos é, por exemplo, a campanha frutícola 2017/2018.

3.3.2.2 Regiões

As regiões representam, de uma forma genérica, polígonos ou multi-polígonos que são definidos através de uma componente geográfica vetorial. Cada região tem um tipo associado, permitindo a representação de diferentes tipos de regiões no sistema. As regiões foram ainda modeladas de forma a permitir qualquer tipo de relação topológica entre duas regiões através da tabela de relações topológicas. Esta tabela de relações topológicas tem uma tabela de tipos associada que para além de definir o tipo de relação, permite também controlar a cardinalidade desta. Um exemplo da aplicação destas relações topológicas é a representação de hierarquia entre regiões. No projeto FitoAgro isso traduz-se, por exemplo, na representação da relação entre parcelas e sub-parcelas (as sub-parcelas são regiões contidas dentro da região principal parcela).

3.3.2.3 Regiões em Períodos

Regiões em períodos permitem representar um dos conceitos principais deste modelo de dados: uma dada região numa dada janela temporal. Este conceito surge através de uma relação N:M entre regiões e períodos e é essencial para agrupar tudo o que tenha ocorrido numa região ao longo de um certo período temporal. Concretizando, uma parcela agrícola ao longo da campanha frutícola 2017/2018 é um exemplo de uma região em período no contexto do projeto.

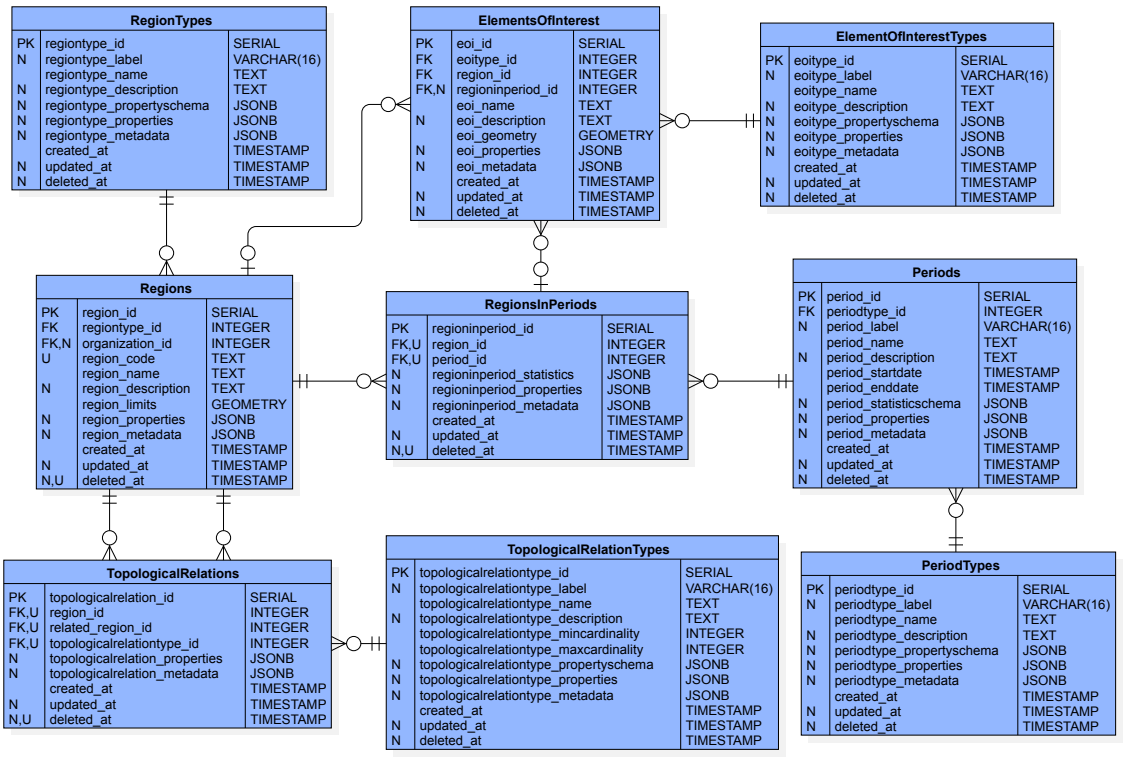


Figura 3.3: Módulo Espaço-Temporal

3.3.2.4 Elementos de Interesse

Elementos de Interesse traduzem-se como sendo a representação de elementos de interesse dentro de uma região, referenciados geograficamente através de dados do tipo vetorial. Estes elementos podem ser divididos em duas categorias: elementos presentes numa região que são tipicamente fixos e não mudam ao longo do tempo (possuem referência para uma região) e elementos presentes numa região que por norma mudam ao longo do tempo (possuem referência para uma região e para uma região em período). São exemplos de tipos de elementos de interesse para o projeto FitoAgro tanto elementos de orientação na parcela (ponto de entrada na parcela, charcos, barracões) como elementos onde são recolhidas observações de campo (árvores, armadilhas). Retomando os exemplos referidos anteriormente, o ponto de entrada na parcela é um caso em que o elemento de interesse é fixo e por outro lado as armadilhas são elementos que apresentam um carácter dinâmico devido a tipicamente mudarem de campanha em campanha.

3.3.3 Módulo de Agro-Negócio

O módulo de Agro-Negócio é um módulo mais específico que modela conceitos mais particulares referentes ao domínio agrícola. Na figura 3.4 pode ser visualizado o diagrama de tabelas deste módulo.

3.3.3.1 Culturas, Variedades e Estados Fenológicos

As variedades refletem variedades agrícolas que podem estar plantadas numa região. Cada variedade pertence a uma cultura específica, que está devidamente tipificada, e cada cultura tem os seus respectivos estados fenológicos. Como para variedades diferentes de culturas iguais, os estados fenológicos apresentam durações diferentes foi criada uma tabela que guarda essa informação. Essa tabela resulta de uma relação N:M entre as variedades e os estados fenológicos.

3.3.3.2 Organismos

Os organismos modelam aquilo que se pretende monitorizar numa região em período e/ou que se querem prever através de modelos preditivos. Estes organismos têm um tipo associado que para o FitoAgro se traduz tipicamente em pragas ou auxiliares. A Cecidómia e a Filoxera são exemplos de dois organismos que se encaixam no tipo praga.

3.3.3.3 Modelos Preditivos

Os modelos preditivos representam de forma genérica os modelos que serão executados para prever a ocorrência das pragas. Para além do seu nome e de uma descrição opcional, possui ainda informação relativa aos seus parâmetros, algo relevante para a operacionalização da sua execução.

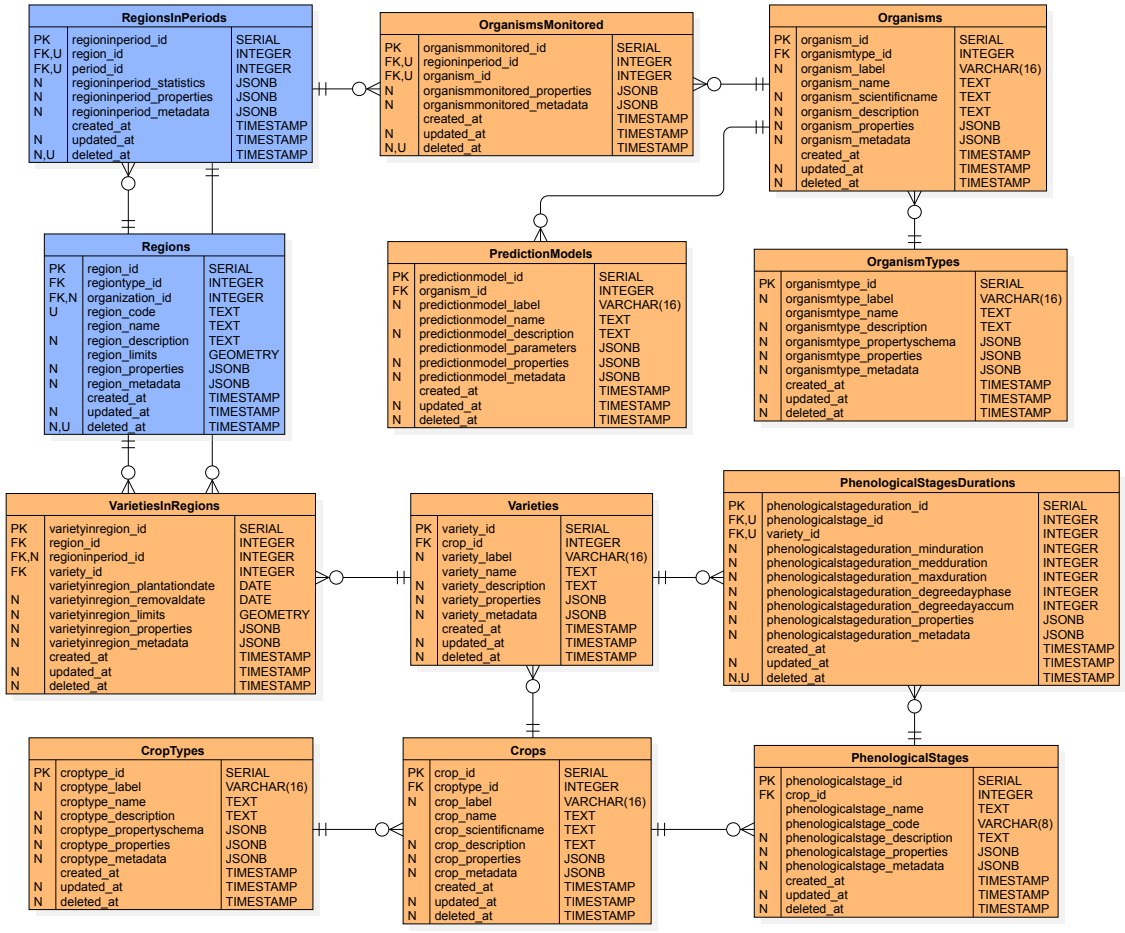


Figura 3.4: Módulo de Agro-Negócio

3.3.4 Módulo de Dados de Campo

O módulo acima referido será o local no modelo de dados onde residirá toda a informação adquirida por um ou mais agentes *in situ*, que no contexto do projeto FitoAgro se traduz em informação recolhida numa parcela agrícola. Na figura 3.5 pode observar-se o diagrama de tabelas deste módulo.

3.3.4.1 Protocolos

Os protocolos traduzem-se de forma genérica como um conjunto de instruções (tipos de observações), definidas por especialistas, a serem executados em regiões em períodos. No caso do projeto FitoAgro, os protocolos existem tipicamente para registar a ocorrência de inimigos e são exemplos disso o protocolo da Cecidómia ou o protocolo da Cochonilha-Algodão.

3.3.4.2 Visitas, Monitorizações e Observações

No contexto de uma dada região em período poderão ser efetuadas visitas que serão representadas na respetiva tabela. Uma visita corresponde à ida a uma dada região com o objetivo de executar uma ou mais monitorizações. Uma monitorização refere-se à execução de um protocolo por um ou mais agentes, da qual resulta um conjunto de observações. As observações têm um tipo associado que indica detalhes como o inimigo em observação (se aplicável), o tipo de dados recolhidos, a sua unidade e ainda atributos próprios para validação e o valor da observação em si.

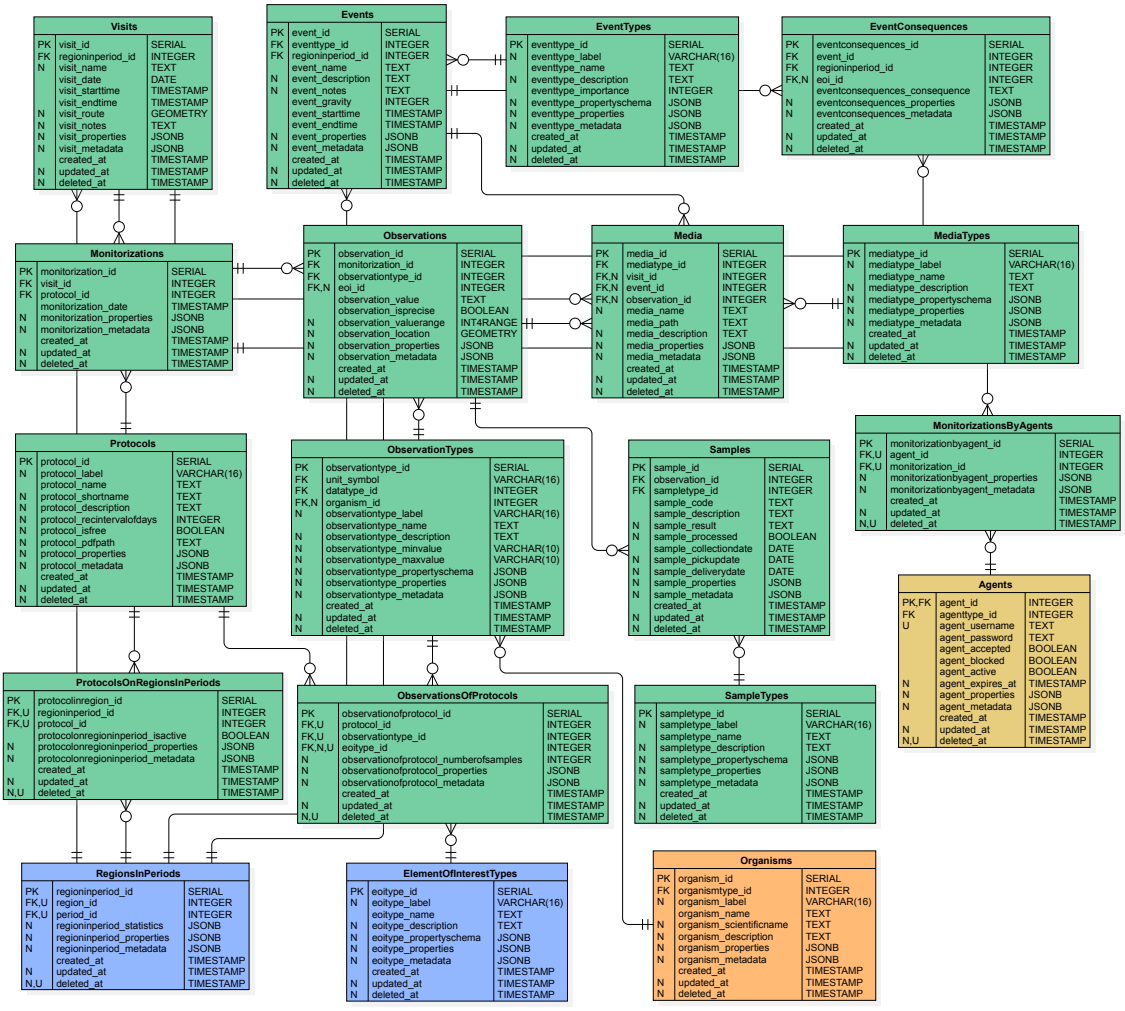


Figura 3.5: Módulo de Dados de Campo

3.3.4.3 Eventos

Os eventos têm como objetivo modelar qualquer tipo de eventos que ocorram numa dada região em período e que sejam considerados relevantes para quem os está a registar ou para o projeto que esteja a fazer uso deste modelo. Este registo terá tipicamente um nome que identifique o evento, a estampilha temporal do início e do fim do evento e a sua gravidade.

Caso haja necessidade é também possível reportar consequências de eventos, que podem ser gerais ao nível da região ou ir em detalhe até ao nível dos elementos de interesse dessa mesma região.

3.3.4.4 Multimédia

Associado a uma visita, a um evento ou a uma observação pode desejar-se guardar conteúdo multimédia (como por exemplo fotografias, vídeos, entre outros). No entanto, devido a este tipo de conteúdo poder atingir dimensões grandes, tanto em quantidade como em espaço nos projetos que este modelo procura cobrir, optou-se por não o guardar directamente na base de dados mas sim por via do *File System*. Na tabela Multimédia estará guardado meramente um apontador para o ficheiro multimédia (um path do *File System*) e o contexto em que este foi recolhido.

3.3.4.5 Amostras

Pode haver a necessidade de recolher amostras dos mais diversos tipos no contexto de uma observação com o objetivo de posteriormente ser analisada em laboratório. Esta entidade procura modelar isso mesmo, permitindo guardar o estado corrente da amostra e o resultado da sua análise. Placas armadilha ou frutos são exemplo de amostras que podem ser recolhidas.

3.3.5 Módulo de Séries Temporais

Representado pela cor vermelha temos o módulo do modelo de dados que estrutura a maneira como as séries temporais provenientes das mais diversas fontes são guardadas. Uma série temporal corresponde a um conjunto de observações de uma variável feitas sequencialmente ao longo do tempo em intervalos uniformes. Na figura 3.6 pode visualizar-se o diagrama de tabelas deste módulo.

3.3.5.1 Fornecedores de Dados

Os fornecedores de dados representam as entidades que fornecem acesso às fontes das séries temporais a serem guardadas no sistema de informação. Para o projeto FitoAgro são exemplos de fornecedores o IPMA, WiseCrop, TerraPro, entre outros.

3.3.5.2 Fontes de Dados

As fontes de dados são de onde as séries temporais provêm, onde estas são geradas. Estas fontes têm associado a si um fornecedor, o seu tipo (que pode ser por exemplo estação meteorológica), as coordenadas geográficas (caso seja uma fonte fixa) e um identificador externo, definido pelo fornecedor e tipicamente utilizado para a aquisição dos dados junto desse fornecedor. Devido a se quererem modelar também fontes virtuais cujas séries sejam, por exemplo, interpoladas através das séries de fontes reais, criou-se uma tabela de

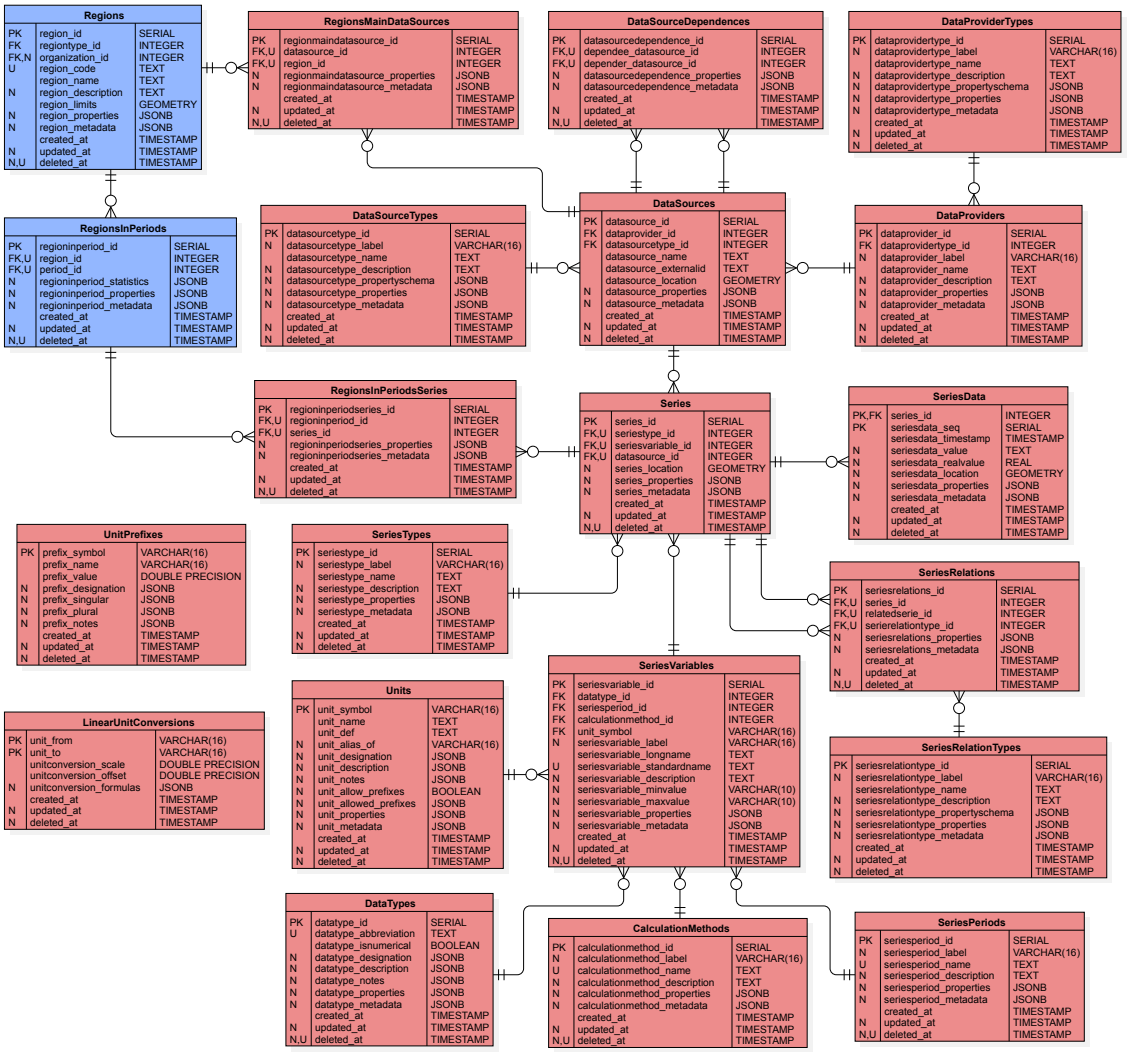


Figura 3.6: Módulo de Séries Temporais

dependências de fontes de dados onde são definidas quais fontes reais são utilizadas para interpolar cada fonte virtual.

3.3.5.3 Identificadores de Séries

Identificadores de Séries são o que definem e identificam o que está a ser registado numa dada série temporal. Estas seguem o *standard* proposto pelo METNorway, referido em 2.3.3, e são por exemplo max(air_temperature PT1H) - que se traduz em temperatura máxima do ar medida de hora a hora. Desse modo são compostas essencialmente por quatro componentes: o seu tipo de dados (como air_temperature), o período em que a série é medida (como PT1H que representa de hora a hora), o seu método de cálculo (como max) e a sua unidade (como ºC).

3.3.5.4 Séries e Dados das Séries

Séries definem as séries temporais propriamente ditas: o que está a ser registado (identificador de série), o que está a gerá-las (fonte de dados) e o seu tipo (tipo de série) que podem ser por exemplo raw ou derivadas. No caso de ser uma série derivada, a informação das séries que são usadas para a derivar fica guardada na tabela de derivação de séries. Existe também a possibilidade de ter uma componente geográfica associada para no caso da fonte de dados não ser fixa.

Na tabela de intitulada de dados das séries é irão residir os valores observados em cada série temporal, tendo também a possibilidade de ter uma componente geográfica associada.

3.3.6 Módulo de Agentes e Organizações

O módulo de agentes e organizações será responsável por conter toda a informação relevante tanto para autenticação como para autorização de todos os agentes que utilizarem o sistema de informação. Na figura 3.7 pode observar-se o diagrama de tabelas deste módulo.

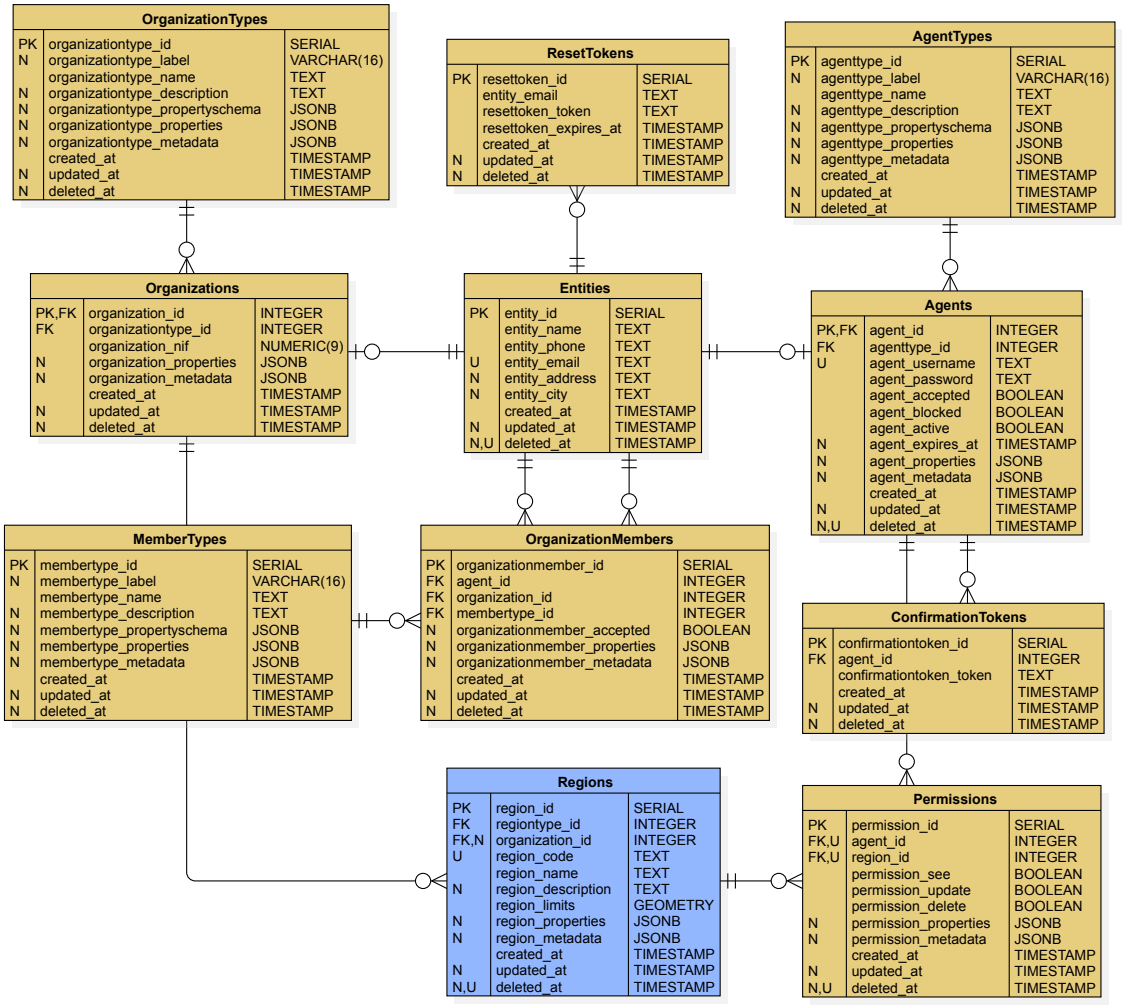


Figura 3.7: Módulo de Agentes e Organizações

3.3.6.1 Organizações

Organizações representam instituições que podem ter algum tipo de associação a agentes do sistema e/ou responsabilidade por regiões. No contexto do projeto FitoAgro são organizações como o COTHN, a APAS e as cooperativas agrícolas que são parceiras do projeto.

3.3.6.2 Agentes

Quando se fala em agentes é o equivalente a falar em alguém ou algo que interage com o sistema. Os agentes têm um tipo associado que servirá para processos de autorização aos recursos do sistema e um conjunto de *flags* booleanas que indicam o seu estado no mesmo. Para o projeto existem agentes como os técnicos de campo, investigadores, administradores, entre outros. Em particular os utilizadores do sistema são agentes.

3.3.6.3 Entidades

As entidades correspondem a uma entidade genérica que tanto pode ser uma organização como um agente. Esta tabela existe com o propósito adicional de guardar os atributos comuns às organizações e aos agentes.

3.3.6.4 Permissões

Permitem definir as acções que cada agente pode fazer à informação referente de cada uma das regiões presentes no sistema.

3.3.7 Módulo de Configurações

Por fim, o módulo de configurações terá informação sobre as configurações base do modelo de dados. Na figura 3.8 pode observar-se o diagrama de tabelas deste módulo.

3.3.7.1 Settings

Settings será onde residirá informação geral sobre o modelo (e.g. versão do modelo).

3.3.7.2 Esquemas de Propriedades

Esquemas de propriedades definirá qual o esquema JSON que será aplicado ao campo 'properties' de cada uma das tabelas.

3.3.7.3 Esquemas de Metadados

Esquemas de metadados definirá qual o esquema JSON que será aplicado ao campo 'metadata' de cada uma das tabelas.

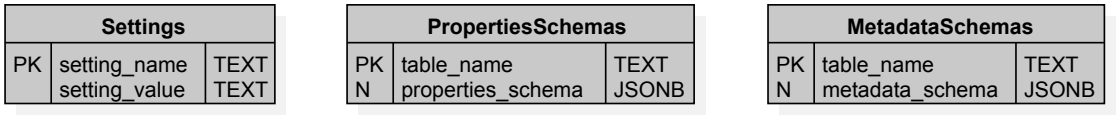


Figura 3.8: Módulo de Configurações

3.4 Data Warehousing

Tal como foi referido em 3.1 o projeto FitoAgro revelou necessidades analíticas que precisavam de ser endereçadas. De forma a responder a esta problemática, foi decidido implementar um Data Warehouse cujo desenho fosse geral mas que ao mesmo tempo permitisse responder aos requisitos analíticos que foram elicitados. Esta secção surge de forma a dar conhecimento, num primeiro ponto, de quais os requisitos que foram tratados e, num segundo ponto, dos modelos multidimensionais que foram desenhados e posteriormente implementados.

3.4.1 Requisitos

No que toca aos requisitos analíticos, estes foram agrupados em duas categorias: controlo da qualidade dos dados e indicadores das pragas. Para o controlo da qualidade dos dados, que tem como *stakeholders* tanto quem recolhe os dados como quem trata da implementação dos processos da integração desses mesmos dados, os requisitos principais são:

- **Quando e onde houve visitas** Perceber a normalidade ou não através da proximidade/afastamento das datas, número de registos abaixo ou acima da média.
- **Revisitar o que aconteceu numa dada visita** Protocolos executados, evolução do estado fenológico, notas, observações e elementos de interesse visitados.
- **Verificação do normal funcionamento das fontes de dados** Data dos últimos dados de cada fonte para poder detetar se há fontes *offline* ou potenciais avarias.
- **Deteção de valores fora do normal** Tanto nas observações de campo como nas séries temporais permitindo detetar avarias de sensores, erros na colheita dos dados e erros na integração dos dados no sistema.

Por outro lado, os indicadores das pragas têm como *stakeholders* os cientistas que estudam as pragas, proporcionando-lhes suporte para a definição dos modelos de previsão. Ao dia de escrita deste documento, a equipa científica ainda não comunicou os respetivos requisitos.

3.4.2 Modelos Multidimensionais

Para dar resposta aos requisitos enunciados em 3.4.1 foram desenvolvidos dois modelos multidimensionais (*star schemas*), que são explicados de seguida.

3.4.2.1 Dados de Campo

Este *star schema*, ilustrado na figura 3.9, tem como objetivo responder aos requisitos analíticos relacionados com os dados recolhidos em campo. O nível de granularidade mais fino possível em que se consegue consultar os dados corresponde a uma observação (o que corresponde a um valor preenchido numa célula de uma folha de registo). De seguida, passa-se a explicar cada uma das dimensões bem como a tabela de factos.

- **Monitorizations**

  A dimensão Monitorizations contém toda a informação que está associada a uma monitorização - protocolo executado, visita em que foi realizada, região e período em que foi aplicada, etc.

- **Observation Types**

  A dimensão Observation Types tem a informação que define o que está a ser observado - tipo de observação, inimigo em questão e ainda informação extra como o tipo de dados ou a unidade do que foi registado.

- **Elements Of Interest**

  A dimensão Elements of Interest define simplesmente, caso exista, o local onde a observação foi realizada.

- **Observations**

  A tabela de factos Observations regista os dados em si. Cada tuplo desta tabela corresponde a uma observação efetuada ou não num elemento de interesse, de um dado tipo, no âmbito de uma monitorização.

3.4.2.2 Séries Temporais

Este outro *star schema*, representado na figura 3.10, tem o objetivo de tratar as necessidades analíticas relacionadas com as séries temporais presentes no sistema. O nível de granularidade mais fino corresponde a uma observação de uma série temporal. Seguidamente são descritas cada uma das dimensões e a tabela de factos.

- **Date**

  A dimensão Date representa e tem informações relacionadas com datas (dia, mês, ano).

- **Time**

  A dimensão Time representa e tem informações relacionadas com tempo (hora, minuto, segundo).

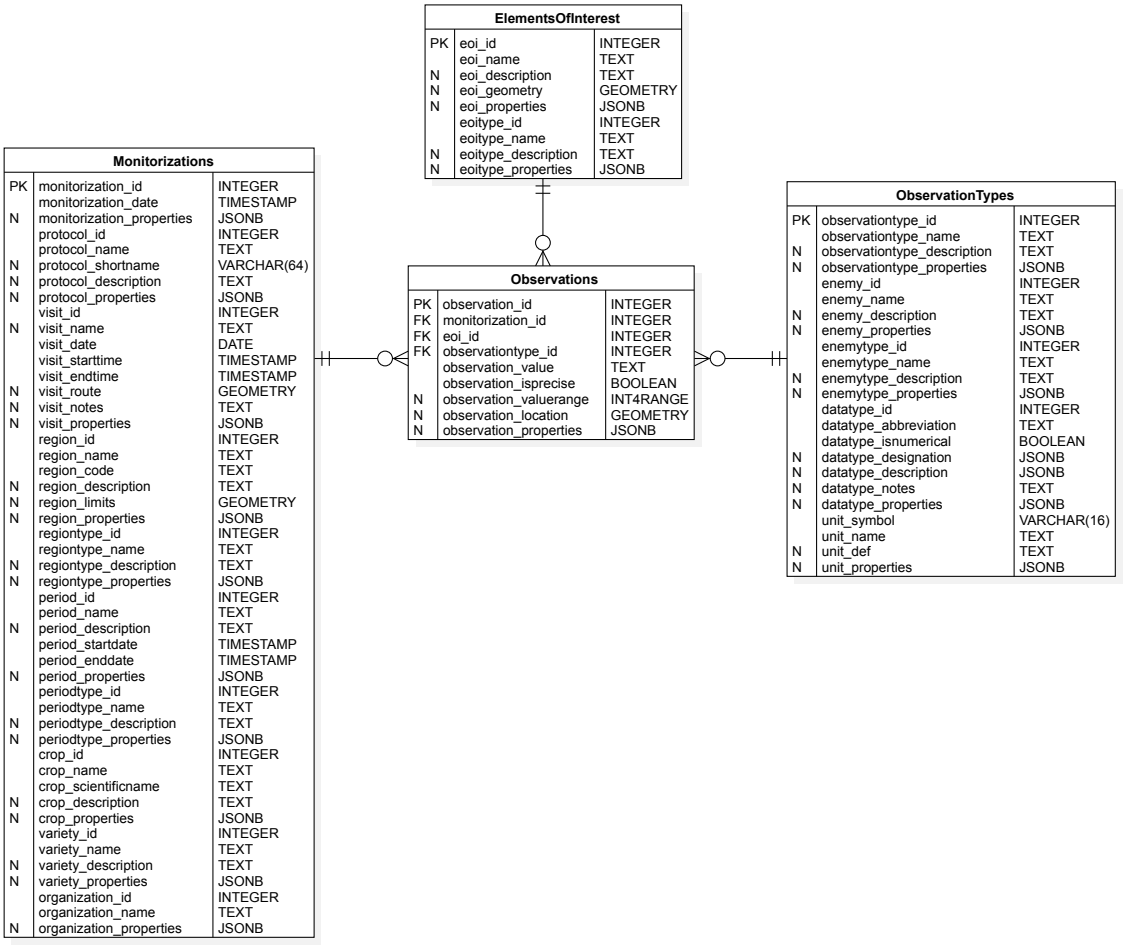


Figura 3.9: *Star schema* dos dados de campo

- **Series Variables**

  A dimensão Series Variables contém toda a informação que permite definir o que está a ser observado numa dada série temporal bem como algumas características desta - intervalo entre observações, tipo de dados, unidade utilizada e método de cálculo.

- **Series Types**

  A dimensão Series Types contém todos os tipos de séries.

- **Data Sources**

  A dimensão Data Sources contém todas as fontes de dados que geram séries temporais e toda a informação que está a si associada - tipo, fornecedor e tipo de fornecedor.

- **Series Data**

  A tabela de factos Series Data possui todos os dados das séries temporais. Simplificando, cada tuplo desta tabela corresponde a uma observação feita numa dada estampilha temporal de uma dada série, de um dado tipo, realizada numa dada fonte.

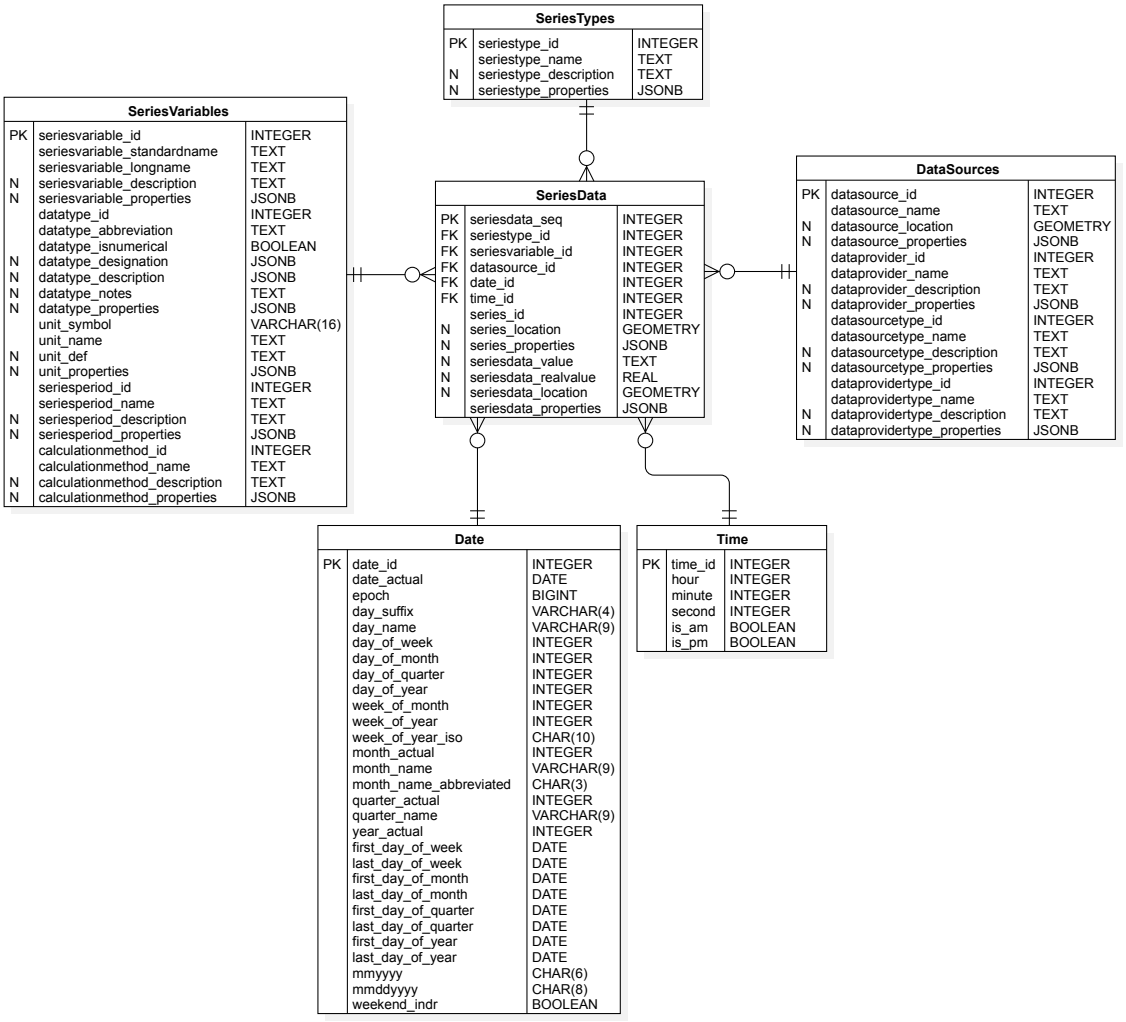


Figura 3.10: *Star schema* das séries temporais

3.5 API REST

De forma a dar acesso a toda a informação presente no sistema de informação à camada Cliente, optou-se por implementar uma API REST. A API REST proporciona a vantagem da camada Cliente e da camada Servidor poderem ser implementadas separadamente, de forma independente, servindo-se apenas desta interface para comunicar entre si. Na presente secção são discutidos três pontos considerados essenciais para o processo de modelação da API, sendo eles: as regras de desenho adotadas, a estrutura da API e os casos de uso que a mesma cobre.

3.5.1 Regras de Desenho

As regras de desenho seguidas para a modelação da API são as seguintes:

- **Modelo de Maturidade Richardson** - A API enquadra-se no nível 2 do modelo de maturidade [53]. Isto significa que utiliza múltiplos recursos - que se traduzem em múltiplos *endpoints* - para disponibilizar a informação. Cada um desses *endpoints* utiliza o método HTTP mais apropriado para a operação em questão. Para além disso, o servidor usa sempre códigos de resposta HTTP para indicar o estado do pedido que foi enviado.

- **Autorização** - A autorização de acesso à API é feita através de um *Bearer Token* passado no *Authorization Header* do pedido. Esse *token* é gerado e enviado para o utilizador após correta autenticação no *endpoint* para esse efeito.

- **Operações CRUD** - Todos os recursos da API disponibilizam cada uma das operações CRUD (Create, Read, Update, Delete) servindo-se dos métodos HTTP POST, GET, PATCH e DELETE respetivamente.

- **Entidades para Recursos** - As entidades do modelo de dados, caso aplicável, foram mapeadas para recursos da API. Foram considerados três tipos distintos de recursos:

    1. Recursos Principais - Recursos considerados independentes o suficiente para existirem por si só.
    2. Recursos Subordinados - Recursos que dependem de outro recurso, isto é, só existem no contexto de outro recurso.
    3. Recursos de Associação - Recursos que resultam da associação entre dois recursos.

- **Querying** - O *querying* aplicado aos pedidos que devolvem coleções de objetos de um dado recurso da API consiste principalmente em filtragem por valor ou, no caso de haver por exemplo datas associadas ao recurso em questão, por *range* de valores. Agregações ou *querying* mais complexo serão suportados pela parte analítica e não por esta API cuja finalidade principal é a operação. Para além da filtragem mencionada, a API faz ainda uso de paginação (utilizando as *keywords* limit e offset) e de ordenação (utilizando as *keywords* sort_by e order_by) para que se possa controlar o *output* de cada resposta.

3.5.2 Estrutura

Dado não existir uma implementação da camada Cliente nem requisitos para a mesma, optou-se por estruturar a API REST de forma a cobrir qualquer operação CRUD que se possa querer aplicar a qualquer recurso. Abaixo é possível ver o mapeamento das entidades do modelo de dados para cada um dos tipo de recursos da API.

Recurso	**Tipo**	**Subordinado de/ Associação entre:**
Agents	Principal	-
Agent Types	Principal	-
Organizations	Principal	-
Organization Types	Principal	-
Organization Members	Associação	Organizations, Agents
Member Types	Principal	-
Permissions	Associação	Agents, Regions
Regions	Principal	-
Region Types	Principal	-
Topological Relations	Associação	Regions
Topological Relation Types	Principal	-
Elements of Interest	Subordinado	Regions
Element of Interest Types	Principal	-
Crops	Principal	-
Crop Types	Principal	-
Varieties	Subordinado	Crops
Phenological Stages	Subordinado	Crops
Phenological Stages Durations	Associação	Varieties, Phenological Stages
Varieties in Regions	Associação	Varieties, Regions
Organisms	Principal	-
Organism Types	Principal	-
Prediction Models	Subordinado	Organisms
Organisms Monitored	Associação	Organisms, Regions In Periods
Periods	Principal	-
Period Types	Principal	-
Regions in Periods	Associação	Regions, Periods
Visits	Subordinado	Regions in Periods
Monitorizations	Subordinado	Visits
Monitorizations by Agents	Associação	Agents, Monitorizations
Observations	Subordinado	Monitorizations
Observation Types	Principal	-
Protocols	Principal	-
Protocols on Regions in Periods	Associação	Protocols, Regions In Periods
Observation of Protocols	Associação	Protocols, Observation Types
Samples	Principal	-
Sample Types	Principal	-
Events	Subordinado	Regions In Periods

Event Types	Principal	-
Event Consequences	Subordinado	Events
Media	Principal	-
Media Types	Principal	-
Data Providers	Principal	-
Data Provider Types	Principal	-
Data Sources	Subordinado	Data Providers
Data Source Types	Principal	-
Data Source Dependences	Associação	Data Sources
Region Main Data Sources	Associação	Regions, Data Sources
Series	Subordinado	Data Sources
Series Types	Principal	-
Series Relations	Associação	Series
Series Relation Types	Principal	-
Regions in Periods Series	Associação	Regions in Periods, Series
Series Variables	Principal	-
Calculation Methods	Principal	-
Series Periods	Principal	-
Data Types	Principal	-
Units	Principal	-

Tabela 3.1: Mapeamento dos recursos da API

Toda a estrutura base da API pode ser gerada através do *template* de cada um dos três tipos de recursos. Existem ainda outros *endpoints* específicos que foram implementados de forma a suportar certos serviços necessários - como é exemplo do *endpoint* /api/token onde o utilizador se autentica para receber o *token* de acesso. Cada um desses *templates* é detalhado de seguida.

Para recursos principais:

- GET api/*recurso* – Retorna a coleção de objetos *recurso*.
- GET api/*recurso*/{*recurso_id*} – Retorna o objeto *recurso* com o identificador *recurso_id*.
- POST api/*recurso* – Cria um objeto *recurso*.
- PATCH api/*recurso*/{*recurso_id*} – Edita o objeto *recurso* com identificador *recurso_id*.
- DELETE api/*recurso*/{*recurso_id*} – Elimina o objeto *recurso* com o identificador *recurso_id*.

Para recursos subordinados:

- GET api/*recurso*/{*recurso_id*}/*subordinado* – Retorna a coleção de objetos *subordinado* do *recurso* com o identificador *recurso_id*.
- GET api/*recurso*/{*recurso_id*}/*subordinado*/{*subordinado_id*} – Retorna o objeto *subordinado* com o identificador *subordinado_id* do *recurso* com o identificador *recurso_id*.
- POST api/*recurso*/{*recurso_id*}/*subordinado* – Cria um objeto *subordinado* do *recurso* com o identificador *recurso_id*.
- PATCH api/*recurso*/{*recurso_id*}/*subordinado*/{*subordinado_id*} – Edita o objeto *subordinado* com o identificador *subordinado_id* do *recurso* com o identificador *recurso_id*.
- DELETE api/*recurso*/{*recurso_id*}/*subordinado*/{*subordinado_id*} – Elimina o objeto *subordinado* com o identificador *subordinado_id* do *recurso* com o identificador *recurso_id*.

Para recursos de associação:

- GET api/*recurso*/{*recurso_id*}/*associação* – Retorna a coleção de objetos *associação* do *recurso* com o identificador *recurso_id*.
- GET api/*associação*/{*associação_id*} – Retorna o objeto *associação* com o identificador *associação_id*.
- POST api/*associação* – Cria um objeto *associação*.
- PATCH api/*associação*/{*associação_id*} – Edita o objeto *associação* com o identificador *associação_id*.
- DELETE api/*associação*/{*associação_id*} – Elimina o objeto *associação* com o identificador *associação_id*.

3.5.3 Casos de Uso

Segue abaixo uma lista dos casos de uso que a API modelada cobre. Devido a ser uma lista extensa, os casos de uso encontram-se generalizados. Como a camada cliente não foi desenvolvida, não serão especificados atores em cada caso de uso. Em vez disso, será assumido um actor genérico que se designará por "utilizador". No apêndice B pode ser consultada a listagem completa dos casos de uso. Os casos de uso genéricos são então:

- O utilizador consulta um recurso.
- O utilizador cria um recurso.

- O utilizador edita um recurso.
- O utilizador elimina um recurso.

Para que se perceba a completude da API, cujos casos de uso possuem múltiplas variantes, será agora exemplificada com um caso de uso que é uma variante do caso C184 do apêndice.

Caso de uso: C184.1

Descrição: O utilizador consulta apenas a sexta e a sétima fonte de dados do fornecedor APAS que são estações meteorológicas, ordenadas alfabeticamente.

Método: GET

Authorization Header: Bearer Token

Endpoint: /api/dataproviders/2/datasources?datasourcetype_id=1&limit=2&offset=5&sort_by=datasource_name&order_by=ASC

Código: 200

Resposta:

Listagem 3.1: Resposta JSON ao caso de uso C184.1


```
[
    {
        "datasource_id": 14,
        "dataprovider_id": 2,
        "datasourcetype_id": 1,
        "datasource_name": "Coopval02AL",
        "datasource_externalid": "Coopval02AL",
        "datasource_location": {
            "type": "Point",
            "coordinates": [
                -9.1261196,
                39.2389178
            ]
        },
        "datasource_properties": null,
        "datasource_metadata": null,
        "created_at": "2019-02-20T16:50:39.787Z",
        "updated_at": null,
        "deleted_at": null
    },
    {
        "datasource_id": 3,
        "dataprovider_id": 2,
        "datasourcetype_id": 1,
        "datasource_name": "Coopval03VN",
```

```
        "datasource_externalid": "Coopval03VN",
        "datasource_location": {
            "type": "Point",
            "coordinates": [
                -9.0943944,
                39.2030109
            ]
        },
        "datasource_properties": null,
        "datasource_metadata": null,
        "created_at": "2019-02-20T16:50:39.772Z",
        "updated_at": null,
        "deleted_at": null
    }
]
```


3.6 Metodologia de Desenvolvimento

Esta secção aborda a metodologia de desenvolvimento que foi definida e adotada ao longo de todo o processo de implementação. A metodologia em questão foi pensada com o objetivo de sistematizar todo o processo de evolução do sistema através de consecutivas versões bem como a sua entrega à máquina em produção.

Deste modo, inicialmente foram definidos três ambientes distintos:

- **Ambiente de Desenvolvimento**

  Ambiente onde são implementadas as *features* da nova versão que estiver em desenvolvimento. Corresponde ao ambiente utilizado pelos desenvolvedores.

- **Ambiente de Testes**

  Ambiente onde são executados testes que validem as novas *features* implementadas no ambiente de desenvolvimento - por um lado verificar o seu correto funcionamento e por outro verificar se não afetam *features* prévias.

- **Ambiente de Produção**

  Ambiente onde a versão mais recente do sistema é executada e disponibilizada aos utilizadores.

Todos os ambientes acima referidos encontram-se interligados através de um repositório Git que trata do controlo de versões. No referido repositório existem três tipos de *branches* que são de seguida explicitados:

- **Feature branches**

  Branches onde são desenvolvidas as novas *features*.

- **Develop branch**

  Branch onde cada *feature branch* é *merged* assim que o seu desenvolvimento seja finalizado. Após todas as *features* novas estejam finalizadas e *merged* no *develop branch*, o ambiente de teste serve-se deste para executar os testes.

- **Master branch**

  Branch onde reside a versão do código a ser executado no ambiente de produção. Uma vez que uma nova versão se encontre testada e pronta para entrega, o *develop branch* é *merged* com este.

O diagrama 3.11 procura ilustrar o fluxo de código entre os três ambientes mencionados e o repositório Git.

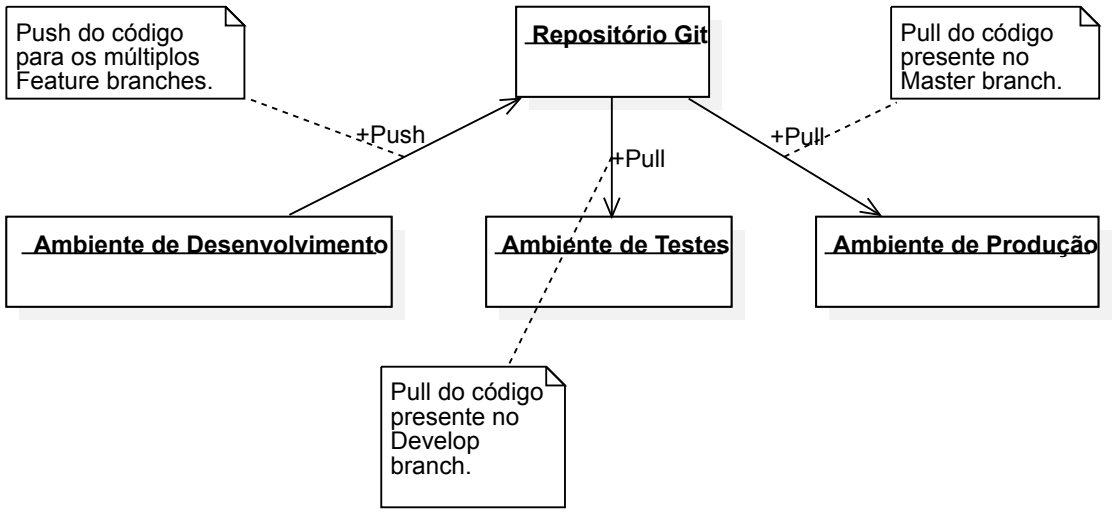


Figura 3.11: Fluxo de código entre ambientes e repositório

3.7 Conclusões

Em suma, foi concebido um modelo de dados tão genérico quanto possível, atendendo tanto à necessidade de adaptabilidade futura a projetos do mesmo domínio como em simultâneo à facilidade da sua utilização. Este modelo cumpre com os seus requisitos, permitindo assim o armazenamento de séries temporais e dados de campo, georreferenciando toda esta informação. Adicionalmente foi ainda modelada uma componente de data warehousing de modo a facilitar as consultas realizadas com propósitos analíticos, cumprindo em simultâneo os requisitos analíticos levantados para o projeto FitoAgro.

No que toca à API REST, esta foi desenhada de forma a permitir consultar e manipular qualquer informação no modelo de dados.

Por fim, a metodologia de desenvolvimento definida e colocada em prática aquando a implementação culminou na sistematização de todo o processo de desenvolvimento - implementação de novas *features*, teste dessa novas *features* e entrega de novas versões do sistema.

Capítulo 4

Implementação

Neste capítulo serão abordados todos os aspetos relacionados com a implementação de todos os componentes do sistema de informação modelado no capítulo anterior.

4.1 Introdução

Após ter sido abordado todo o processo de modelação chegamos ao capítulo que trata de detalhar tudo o que tenha a ver com a implementação do sistema de informação. De notar que tudo o que for abordado ao longo deste capítulo corresponde à versão do sistema de informação que se encontra atualmente em produção, alojado numa máquina virtual cedida pelo Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa.

Inicialmente serão discutidas as tecnologias principais (4.2) que foram utilizadas para a implementação. De seguida e aproveitando desde já para revisitar a arquitetura do sistema, apresentada em 3.2, será explicada a forma como cada um dos componentes desta arquitetura foi implementado:

- Base de Dados Operacional e ORM em 4.3
- API REST e Lógica do Negócio em 4.4
- Integrador de Dados em 4.5 e 4.6
- Base de Dados Analítica em 4.7
- Camada de Visualização em 4.8

Por fim, o presente capítulo termina com uma pequena referência à documentação que foi gerada para o sistema de informação (4.9).

4.2 Tecnologias Escolhidas

Nesta secção serão explicadas as tecnologias escolhidas, exibindo uma breve justificação acerca da sua escolha.

4.2.1 PostgreSQL

A base de dados adotada foi o PostgreSQL. Esta base de dados apresenta-se como sendo a solução *standard* quando se procura uma base de dados relacional *open-source*. Para além disso oferece a extensão PostGIS que dá resposta a todas as necessidades de georreferenciação dos dados.

Apesar de se terem considerado alternativas não relacionais devido à necessidade de representação de séries temporais (exemplo: dados meteorológicos), optou-se por escolher uma base de dados relacional por se considerar que a integridade e consistência dos dados oferecida pelo relacional compensa a perda de escalabilidade (ponto forte das não relacionais) pelo menos no âmbito deste sistema de informação.

Reforçando, tanto o projeto Protomate como o SafeBrócolo utilizaram esta base de dados obtendo bons resultados o que indica que esta opção já provou no passado ser adequada para sistemas de informação deste âmbito.

4.2.2 Node.js

No que toca à tecnologia a utilizar no servidor para implementar a API e a lógica do negócio foi escolhido Node.js, servindo-se da *framework web* minimalista Express.js e do ORM Sequelize.

Para além familiarização com Javascript, o *framework* Express.js permite com a maior das facilidades desenvolver o lado do servidor, suportando a criação tanto da API como, de futuro e caso seja considerado relevante, a criação de uma aplicação web.

No que toca ao ORM optou-se pelo Sequelize por ser o mais popular entre todas as opções disponíveis para Node.

4.2.3 Python

A componente Integrador de Dados da Camada Servidor levantou a necessidade de usar uma linguagem de *scripting* que permitisse uma fácil manipulação desses mesmos dados.

Como tal, Python surge como a solução mais óbvia. Fácil de aprender, bem documentada e com excelente suporte da comunidade são 3 das maiores vantagens que esta linguagem apresenta, sem esquecer as imensas bibliotecas que oferece - em especial bibliotecas como o pandas que são completamente orientadas à manipulação e análise de dados. A componente Integrador de Dados foi totalmente implementada nesta linguagem.

4.2.4 NGINX

NGINX é um servidor HTTP de alta performance, *open-source*, que tem emergido bastante nos últimos ano, ocupando atualmente o segundo lugar no mercado deste tipo de servidores - apenas atrás do Apache. O mesmo foi escolhido para ser utilizado na implementação do sistema de informação por apresentar melhor performance do que o Apache [59] e por todas as vantagens que apresenta quando combinado com Node.js. Essas vantagens serão discutidas mais à frente em 4.4.1.

4.2.5 Tableau

O Tableau foi o software escolhido para implementar toda a Camada de Visualização. Foi escolhido por três razões: das melhores, se não mesmo a melhor, solução no mercado no que toca a ferramentas analíticas, uso livre em projetos de investigação e o conhecimento prévio deste software.

4.3 Base de Dados

A presente secção aborda algumas partes relevantes para aquilo que foi a implementação da base de dados.

4.3.1 Conexão à Base de Dados

A interação do servidor com a base de dados será feita através do Sequelize. Este ORM configura uma *pool* de conexões assim que é inicializado, reutilizando conexões sempre que possível. A configuração da *pool* é totalmente personalizável consoante as necessidades do sistema.

Listagem 4.1: Inicialização do Sequelize

```
1 const sequelize = new Sequelize(config.database, config.username, config.
    ↪ password, {
2     dialect: 'postgres',
3     pool:{
4         max: 5,
5         min: 0,
6         idle: 10000,
7         handleDisconnects: true
8     }
9 });
```

4.3.2 Modelos

Um modelo é uma classe que representa uma tabela da base de dados. É através destas classes que todo o mapeamento das tabelas é realizado. Uma instância de um modelo representa um tuplo da tabela. Em cada uma destas classes é necessário definir a tabela que se está a mapear, declarar todos os atributos com os seus detalhes, podendo ainda definir validações de *inputs* a este nível. Foram assim implementados modelos para todas as tabelas existentes na base de dados.

Listagem 4.2: Modelo para a tabela DataSources

```
let DataSource = sequelize.define('datasources', {
       datasource_id: {
           type: DataTypes.INTEGER,
           allowNull: false,
           primaryKey: true,
           autoIncrement: true
       },
       datasource_name: {
           type: DataTypes.TEXT,
           allowNull: false,
           validate: {
               notEmpty:{
                   msg: 'Data source name should not be empty.'
               },
               notNull:{
                   msg: 'Data source name should be provided.'
               }
           }
       },
       (...)
```

Adicionalmente, os modelos permitem que se definam as relações que existem entre estes, isto é, permitem que se faça o mapeamento das relações existentes entre as tabelas da base de dados que representam (1:1, 1:N, N:M). Dar esta informação ao Sequelize é particularmente relevante caso se queira fazer *eager-loading* em queries. Deste modo, todas as relações existentes entre modelos foram definidas.

Listagem 4.3: Implementação das relações para o modelo DataSource

```
DataSource.associate = function(models) {
       DataSource.hasMany(models.series, {
           as: 'series',
           foreignKey: 'datasource_id'
```

```
        });
        DataSource.belongsTo(models.dataproviders, {
            as: 'data_provider',
            foreignKey: 'dataprovider_id'
        });
        DataSource.belongsToMany(models.datasources, {
            through: 'datasourcedependences',
            as: 'dependee',
            foreignKey: 'dependee_datasource_id'
        });
        (...)
    };
```

Estes modelos disponibilizam nativamente um conjunto de funções a partir do qual são realizadas as interações com a base de dados. Dentro dessas funções são então realçadas abaixo as que mais foram utilizadas ao longo da implementação da API:

- findAll - Procura múltiplas instâncias.
- findByPk - Procura uma instância pela chave-primária.
- create - Cria uma instância.
- update - Atualiza uma instância.
- destroy - Elimina uma instância.

4.3.3 Índices

Visto a base de dados lidar com séries temporais, a dimensão da tabela onde os dados destas residem (SeriesData) tende para crescer bastante ao longo do tempo. Por exemplo, ao dia 6 de Setembro de 2019 a tabela referida já conta com mais de 18 milhões de tuplos.

De forma a permitir o rápido acesso a estes dados, muitas vezes pedidos por intervalos temporais, foi criado o seguinte índice:

```
CREATE INDEX ON seriesdata
  (series_id, seriesdata_timestamp, seriesdata_realvalue);
```

Este índice é usado automaticamente pelo Postgres em consultas que seja considerado vantajoso.

4.3.4 Funções

Por fim, resta referir que foram criadas 3 funções PL//pgSQL que devolvem a informação geográfica das regiões, elementos de interesse e fontes de dados em GeoJSON. Sentiu-se a necessidade de criar estas funções uma vez que o Tableau não consegue interpretar o

tipo de dados Geometry do PostGIS. Este facto inviabilizaria qualquer visualização que se quisesse fazer com o componente geográfico pelo que a maneira de resolver este problema foi criar estas funções. Uma das funções é apresentada na listagem que se segue.

Listagem 4.4: Função para devolver GeoJSON dos limites das regiões

```
1  CREATE OR REPLACE FUNCTION region_limits_as_geojson ()
2  RETURNS JSON AS $$
3  DECLARE regions_geojson JSON;
4  BEGIN
5    SELECT row_to_json(fc) INTO regions_geojson
6    FROM ( SELECT 'FeatureCollection' As type,
7                  array_to_json(array_agg(f)) As features
8          FROM (SELECT 'Feature' As type
9                    , ST_AsGeoJSON(region_limits)::json As geometry
10                   , row_to_json((SELECT l FROM (SELECT region_id,
11                                                       region_name) As l
12                                 )) As properties
13               FROM regions As lg ) As f ) As fc;
14   RETURN regions_geojson;
15 END;
16 $$
17 LANGUAGE plpgsql;
```

4.4 Servidor

A implementação do servidor é detalhada nesta secção. Para um melhor entendimento de como este e a API REST que expõe foram implementados será dado um exemplo de um pedido que seja feito via à API, explicando por onde este passa desde que o cliente faz o pedido até receber a sua resposta. Imaginemos então um cenário em que um utilizador quer listar todas as estações meteorológicas do fornecedor APAS. Este faz então o pedido para: /api/dataproviders/2/datasources?datasourcetype_id=1.

4.4.1 Servidor Web

O pedido ao ser enviado para o servidor é recebido inicialmente por um servidor web NGINX. A decisão de colocar este servidor antes do servidor Node tem diversas motivações baseadas nas vantagens que o uso em conjunto destes traz.

Para começar, pretende-se que todos os pedidos utilizem HTTPS de forma a garantir encriptação entre todas as comunicações entre os clientes e o servidor. Para tratar deste assunto apenas com um servidor Node pode trazer algumas complicações desnecessárias, nomeadamente no que toca a privilégios e atribuições de portas dado ser necessário reservar

a porta 443 (porta HTTPS da máquina em produção). O NGINX resolve este problema ao atuar como um *proxy* reverso: recebe os pedidos que chegam via porta 443 e reencaminha-os para a porta onde estiver o servidor Node, encaminhando depois as respostas provenientes para os clientes. O certificado utilizado foi criado junto da autoridade de certificação Let's Encrypt [35].

Adicionalmente, esta arquitetura permite de forma simples escalabilidade horizontal do sistema. Caso haja um aumento significativo no número de pedidos, basta lançar mais instâncias do servidor Node e indicá-las ao NGINX. Através dos mecanismos de balanceamento de carga que possui, este distribui os pedidos pelas instâncias Node que estiverem em execução.

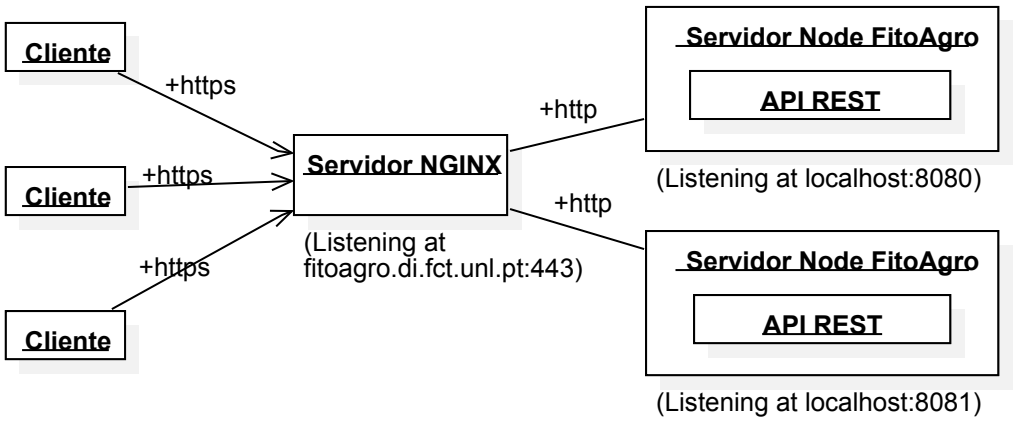


Figura 4.1: *Proxy* reverso e balanceamento de carga do NGINX

4.4.2 Routing

Uma vez o pedido entregue ao servidor Node, este faz o *routing* servindo-se da classe Router disponibilizada pelo Express. É através desta classe que todos os *endpoints* da API são definidos no servidor. O servidor encaminha então o pedido para o *endpoint* correspondente, onde inicialmente verifica o *token* de autorização (mencionado com mais detalhe em 4.4.3) e caso tenha sucesso envia o pedido para o controlador (abordados em 4.4.4) executar a lógica de negócio associada a esse pedido. A listagem seguinte exibe o código responsável por tratar do pedido exemplo.

Listagem 4.5: Lógica do routing da API

```
1 router.get('/dataproviders/:dataprovider_id(\\d+)/datasources',
    ↪ verifyToken, controllers.dataSourcesController.datasource_list);
```

4.4.3 Autenticação e Autorização

Todos os pedidos feitos para a API REST do servidor têm obrigatoriamente de ter no seu *header* um *token* de autorização. Este *token* é fornecido após o utilizador se autenticar

via o *endpoint* para essa finalidade (/api/token). Após o servidor validar a autenticação do utilizador através das suas credenciais passadas no corpo do pedido para o *endpoint* /api/token, gera um JSON Web Token, válido durante 24 horas, que servirá para as subsequentes interações entre o utilizador e a API. O *token* é gerado através do algoritmo HS256.

Listagem 4.6: Lógica de autenticação

```
exports.token = function (req, res, next) {
    // Lógica de autenticaçao
    (...)
    let passwordIsValid = bcrypt.compareSync(req.body.agent_password,
        ↪ agent.agent_password);
    if (!passwordIsValid)
        return next(new createError.Unauthorized('Wrong credentials.'));
    else{
        jwt.sign({agent: {
                        agent_id: agent.agent_id,
                        agenttype_id: agent.agenttype_id,
                        agent_username: agent.agent_username
                       }
                }, 'secretkey', {expiresIn: '24h'}, (err, token) => {
            res.json({
                token
            })
        });
    }
```

Posteriormente, em cada *endpoint* é utilizada a função abaixo demonstrada para validar o *token* e consequentemente autorizar ou não o seu acesso.

Listagem 4.7: Lógica de autorização

```
function verifyToken (req, res, next) {
    const bearerHeader = req.headers['authorization'];
    if (typeof bearerHeader !== 'undefined'){
        // Parse do token
        (...)
        jwt.verify(token, 'secretkey', (err, authData) => {
            if (err) {
                return next(new createError.Forbidden());
            }
            else {
                next();
```

```
            }
        })
    }
    else {
        next(new createError.Forbidden());
    }
}
```

4.4.4 Controladores

Os controladores são o local onde reside a lógica dos *endpoints* da API REST. Implementou-se um controlador para cada recurso da API sendo que este trata de todos os pedidos relacionados com o recurso em questão (uma função para cada endpoint do recurso em questão).

Estes controladores utilizam os modelos do Sequelize (referidos em 4.3.2) que precisarem para a sua lógica e servem-se das operações que estes disponibilizam para fazerem a interação com a base de dados.

Um pedido ao chegar a um controlador é então direcionado para a função que trata do *endpoint* em questão, faz *parse* dos parâmetros do pedido, executa a lógica referente e por fim trata de enviar a resposta.

Considerando o pedido utilizado ao longo desta secção como exemplo, este é entregue à função presente no controlador do recurso Data Source demonstrada na listagem 4.8. De notar que foi adotada a filosofia de *lazy-loading* na implementação de todos os *endpoints* que realizam consultas aos dados. Devido a não existirem requisitos para a Camada Cliente foi pretendido desta forma reduzir ao máximo *payload* desnecessário na resposta de cada pedido.

Na listagem 4.8 é possível observar o código responsável por tratar do pedido utilizado como exemplo, enviando depois a resposta para o cliente.

Listagem 4.8: Exemplo de lógica de um *endpoint*

```
exports.datasource_list = function(req, res, next){

    // Parse dos parâmetros do pedido
    (...)
    models.datasources.findAll({
        attributes: ['datasource_id', 'datasource_name', '
            ↪ datasource_externalid'],
        where: whereStatement,
        order: order,
        limit: limit,
        offset: offset
```

```
11      }).then(datasources =>{
12          res.json(datasources);
13      }).catch(function (err){
14          return next(new createError.BadRequest(err.message));
15      })
16
17  };
```

4.5 Integração dos Dados Meteorológicos

Os dados meteorológicos integrados no sistema são essencialmente de dois tipos: dados de observações meteorológicas e dados de previsão meteorológica. Estes apresentam imensa heterogeneidade, tanto na forma de acesso como no formato em que são disponibilizados. Em 4.1 pode-se observar-se uma tabela que procura exibir essa mesma heterogeneidade.

Fornecedor	**Acesso**	**Formato**
APAS	Dropbox	.txt, .wlk
IPMA	API	.json
PESSL/FieldClimate	API	.json
TerraPro	SFTP	.xml
WiseCrop	API	.xlsx

Tabela 4.1: Forma de acesso e formato por fornecedor

Para que se perceba a extensão destas fontes estamos a falar de 59 fontes de dados distintas, repartidas da seguinte forma pelos fornecedores:

- **APAS**: 14 estações meteorológicas.
- **IPMA**: 7 estações meteorológicas + previsões meteorológicas de 19 locais.
- **PESSL - FieldClimate**: 10 estações meteorológicas.
- **TerraPro**: 3 estações meteorológicas.
- **WiseCrop**: 6 estações meteorológicas.

Devido a toda esta heterogeneidade e também ao facto de a qualquer momento se poder querer adicionar novos fornecedores ou novas fontes de dados, a integração foi pensada e implementada de uma forma genérica recorrendo a dois módulos principais, o módulo Fetcher e o módulo Data Processor, que são descritos nas secções 4.5.2 e 4.5.3.

4.5.0.1 Formatos

Até serem inseridos no sistema, os dados meteorológicos passam por 3 formatos distintos:

1. **Formato Raw** - Formato que corresponde àquele em que os dados são disponibilizados pelo fornecedor. É o formato recebido pelos Fetchers. Um ficheiro por cada fonte.

2. **Formato Intermédio** - Formato CSV a partir do qual os dados passam a estar num formato unificado, isto é, igual para todos os fornecedores. Será gerado no *output* dos Fetchers e é o formato esperado pelos Data Processors. Um ficheiro por cada fonte.

3. **Formato Final**: Formato CSV que coincide com o que é utilizado para a inserção na base de dados. É o *output* final dos Data Processors. Um ficheiro por fornecedor.

Figura 4.2: Formatos dos dados ao longo da integração

Os dados são sempre guardados em cada um dos 3 formatos no File System respeitando a seguinte estrutura de pastas concretizada com um exemplo:

```
data
├── csv_final
│   └── WiseCrop
│       └── 2019-08-30.csv
├── csv_inter
│   └── WiseCrop
│       └── 294
│           └── 2019-08-30.csv
└── raw
    └── WiseCrop
        └── 294
            └── 2019-08-30.xlsx
```

A decisão de guardar todos os passos foi tomada com o racional de servir para histórico e também para não se perder o resultado de nenhum *output* tanto do Módulo Fetcher como do Módulo Data Processor.

4.5.0.2 Previsões Meteorológicas

A integração das previsões meteorológicas é efetuada de forma igual às observações meteorológicas, sendo que o locais das previsões são tratados como se fossem uma fonte de dados, ainda que virtual. Em relação aos dados de previsão, à medida que novas previsões são inseridas, previsões anteriores são sempre substituídas pelas mais recentes. Desta forma, qualquer consulta realizada a estes dados devolve sempre as últimas previsões.

4.5.1 Processo Master

Um processo Master corresponde, de forma simples, a um processo que inicializa e executa as instâncias do módulo Fetcher e do módulo Data Processor de cada um dos fornecedores. Na máquina em produção foi definido um cronjob que executa diariamente o processo Master implementado. Optou-se por fazer a integração diariamente por dois motivos: não existe, para este projeto, a necessidade de ter os dados em *near real time* e a frequência de publicação dos dados por parte dos fornecedores é relativa ao próprio (temos o exemplo da APAS cuja frequência varia de 7 a 8 horas e o exemplo da WiseCrop cuja frequência de publicação está entre os 15 e os 30 minutos).

De forma a melhorar a performance do processo de integração destes dados foi feito o uso de threading. A integração foi separada ao nível do fornecedor pelo que se criou uma função para cada um. Cada função instancia o fetcher e o data processor correspondentes. Posteriormente foi criada uma *thread* por cada função.

Abaixo é possível ver parte do código do processo Master implementado.

Listagem 4.9: Script master.py

```
def wisecrop():
    fetcher_wisecrop.fetch(email='sample@sample.com', password='samplepw')
    wisecrop_processor = dataprocessor.DataProcessor(
        provider='WiseCrop',
        sources_config_dir=os.path.join(file_dir, 'sources_config/WiseCrop'
            ↪ ),
        csv_inter=os.path.join(file_dir, 'data/csv_inter/WiseCrop'),
        csv_final=os.path.join(file_dir, 'data/csv_final/WiseCrop'),
        localhost=config['host'],
        db_name=config['database'],
        db_username=config['username'],
        db_password=config['password']
    )
    wisecrop_processor.process(
        sources=['294', '612', '642', '643', '697'],
        transformations={
       'Velocidade␣do␣Vento␣(Km/h)␣': lambda x: round(x / 3.6, 1),
       'Rajada␣de␣Vento␣(Km/h)␣': lambda x: round(x / 3.6, 1),
       'Folha␣Molhada␣()␣': lambda x: x * 15
       }
   )
    wisecrop_processor.process(
       sources=['717'],
       transformations={
       'Velocidade␣do␣Vento␣(Km/h)␣': lambda x: round(x / 3.6, 1),
```

```
25        'Rajada␣de␣Vento␣(Km/h)␣': lambda x: round(x / 3.6, 1)
26        }
27    )
28     wisecrop_processor.integrate()
29
30 threading.Thread(target=apas).start()
31 threading.Thread(target=ipma).start()
32 threading.Thread(target=pessl).start()
33 threading.Thread(target=terrapro).start()
34 threading.Thread(target=wisecrop).start()
```

4.5.2 Módulo Fetcher

O módulo Fetcher consiste num conjunto de *scripts* Python, denominados Fetchers, cujo seu objetivo é a aquisição dos dados e a unificação do seu formato. Devido às particularidades de cada fornecedor, já referidas anteriormente, foi realizada a implementação de um Fetcher específico para cada um.

Existem dois tipos de Fetchers tendo como diferença principal o método de aquisição dos dados que pode ser via Pull (4.5.2.1) ou via Push (4.5.2.2).

O resultado final da execução de ambos os tipos de Fetcher é um ficheiro CSV no formato intermédio por cada fonte do fornecedor em questão. Este formato consiste numa primeira coluna que indica a estampilha temporal e uma coluna por cada série temporal que a fonte estiver a gerar. A tabela 4.2 exemplifica o formato intermédio.

Data/Hora	**Temperatura (ºC)**	**Humidade (%)**	**Pluviosidade (mm)**
2019-08-30 00:00	17.22	98.74	0
2019-08-30 00:15	17.33	98.98	0
2019-08-30 00:30	17.29	99.1	0
2019-08-30 00:45	17.26	99.1	0
2019-08-30 01:00	17.22	98.95	0

Tabela 4.2: Exemplo do formato intermédio - fragmento extraído do ficheiro de dia 30-08-2019 da estação 294 do fornecedor WiseCrop

4.5.2.1 Pull Fetcher

No projeto FitoAgro a maioria dos Fetchers implementados correspondem a Pull Fetchers. Estes Fetchers são então responsáveis por pedir os dados ao fornecedor em questão para depois os passarem para o formato intermédio.

Em 4.3 é possível ver o diagrama de atividades que explicita os passos genéricos que um Pull Fetcher realiza quando é executado.

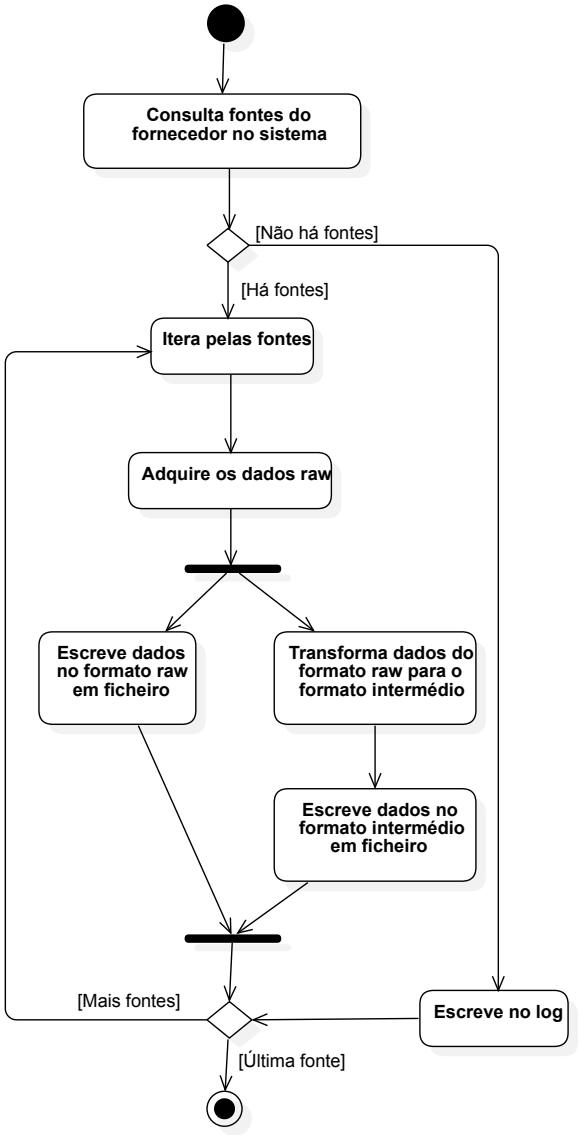


Figura 4.3: Diagrama de atividades de um Pull Fetcher

4.5.2.2 Push Fetcher

Um Push Fetcher é em tudo semelhante a um Pull Fetcher excetuando a parte de aquisição dos dados. Enquanto que um Pull Fetcher pede ao fornecedor os dados, num Push Fetcher é o próprio fornecedor que coloca os dados do nosso lado.

Exemplificando, no projeto FitoAgro apenas o fornecedor TerraPro necessita de um Push Fetcher. Foi criado para este fornecedor um utilizador SFTP a partir do qual este tem acesso à diretoria onde coloca os dados das suas fontes. Esta diretoria tem uma pasta para cada uma das fontes.

Para um Push Fetcher na parte de aquisição dos dados só é preciso indicar a diretoria onde estão a ser colocados os dados. Toda a outra parte de transformação do formato raw para o formato intermédio é similar ao Pull Fetcher.

Em 4.4 é possível ver o diagrama de atividades que explicita os passos genéricos que um Push Fetcher realiza quando é executado.

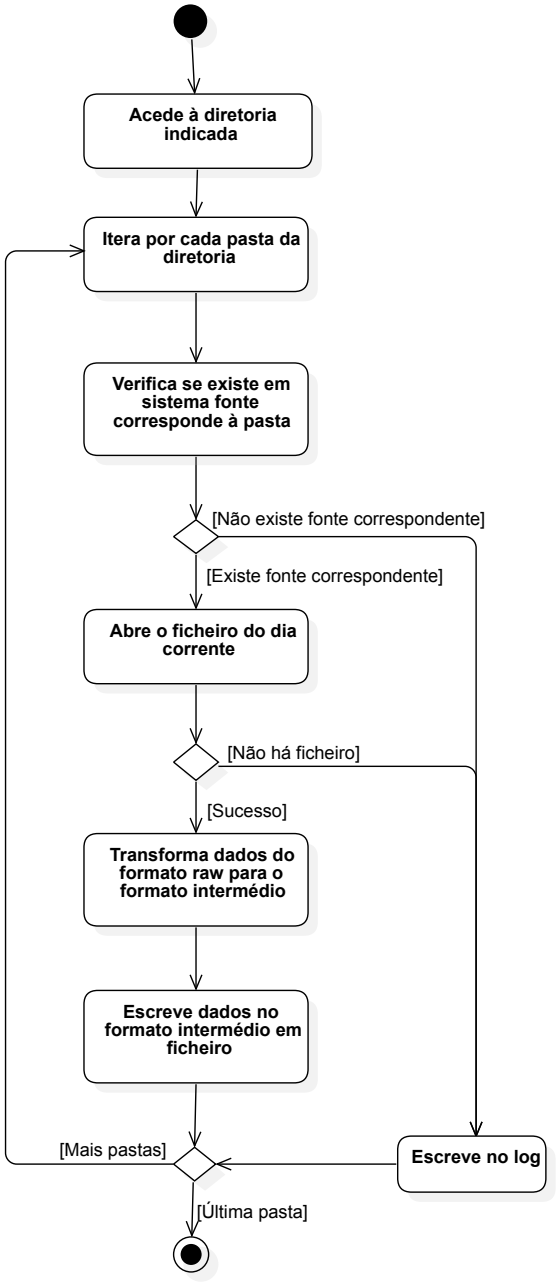


Figura 4.4: Diagrama de atividades de um Push Fetcher

4.5.3 Módulo Data Processor

Por fim, o módulo Data Processor corresponde a um módulo genérico que, recebendo os dados no formato intermédio, é responsável por realizar todas as transformações/substituições e mapeamentos nos dados, servindo-se das funcionalidades da biblioteca pandas do

Python, antes de os passar para o formato final e os inserir na base de dados do sistema de informação.

Este módulo é instanciado passando argumentos que definem: o fornecedor, a diretoria onde estão os dados no formato intermédio, a diretoria onde guardar os dados no formato final, a diretoria onde estão os ficheiros de configuração de cada uma das fontes de dados e as configurações da base de dados.

Uma vez instanciado disponibiliza duas funções: a função process e a função integrate. A função process é aquela que trata da aplicação das transformações/substituições e mapeamentos, passando os dados para o formato final. Esta recebe como argumentos opcionais as transformações (um dicionário que diz para cada coluna qual transformação aplicar aos valores, exemplo: conversões de unidades), as substituições (um dicionário que diz para cada coluna qual o conjunto de substituições de valores se deve fazer, exemplo: substituir pontos cardeais na direção do vento pelo grau correspondente) e as fontes de dados que vai processar (este último teve como racional dar a possibilidade de aplicar transformações e substituições diferentes a fontes do mesmo fornecedor).

A função integrate é a que pega no ficheiro em formato final e o insere na base de dados recorrendo ao comando COPY do PostgreSQL, limpando depois os dados anteriores que tenham sido corrigidos nesta última inserção. Optou-se por utilizar o COPY na inserção dos dados para maximizar a eficiência do processo dado este inserir todos os valores de uma só vez.

Na figura 4.5 é possível ver o diagrama de atividades que explicita os passos genéricos que um Data Processor realiza quando é executado.

4.5.3.1 Configuração das Fontes

De forma a fazer o mapeamento dos identificadores das séries dos fornecedores para os identificadores das séries do sistema de informação foi necessário definir ficheiros de configuração.

Devido à configuração das fontes de dados tipicamente variarem entre si (até em fontes do mesmo fornecedor), definiu-se um ficheiro de configuração por fonte.

Os ficheiros de configuração correspondem a um JSON que permite fazer o mapeamento dos *headers* das colunas da séries temporais presentes no formato intermédio. Na listagem 4.10 é possível ver um exemplo de ficheiro de configuração para uma fonte de dados.

Listagem 4.10: JSON de configuração da estação 294 do fornecedor WiseCrop

```
1 {
2     "Temperatura␣(C)␣": "mean(air_temperature␣PT15M)",
3     "Humidade␣(%)␣": "mean(relative_humidity␣PT15M)",
4     "Pluviosidade␣(mm)␣": "sum(precipitation_amount␣PT15M)",
5     "Radiaçao␣(W/m2)␣": "mean(solar_irradiance␣PT15M)",
6     "Velocidade␣do␣Vento␣(Km/h)␣": "mean(wind_speed␣PT15M)",
```

```
7     "Direçao␣do␣vento␣(␣)␣": "mean(wind_from_direction␣PT15M)",
8     "Rajada␣de␣Vento␣(Km/h)␣": "max(wind_speed␣PT15M)",
9     "Folha␣Molhada␣(␣)␣": "accumulated(leaf_wetness␣PT15M)",
10    "VPD␣(kPa)␣": "mean(water_vapor_saturation_deficit␣PT15M)"
11 }
```

4.5.4 Logging

O processo de integração destes dados resulta então na execução simultânea de múltiplos processos Fetcher e Data Processor chamadas pelo processo principal Master. Pelo que foi referido, é importante que haja uma maneira de registar os eventos relevantes que ocorreram ao longo destes processos, contribuindo diretamente para o requisito do sistema ser auditável. Optou-se então pela implementação, através do módulo Logging do Python, de um log comum a todos os processos referidos anteriormente que será inicializado em cada um da seguinte maneira:

Listagem 4.11: Inicialização do logger

```
1 logging.basicConfig(level=logging.INFO,
2                     format='%(asctime)s|%(name)s|%(levelname)s|%(message)s',
3                     filename=file_dir + '/logs/%s.log' % datetime.datetime.
                          ↪ now().date(),
4                     filemode='a')
5 logger = logging.getLogger(__name__)
```

A estrutura do log é constituída por 4 elementos:

- **Estampilha Temporal** - Quando o evento ocorreu.
- **Nome do Processo** - Processo onde o evento ocorreu.
- **Nível da Mensagem** - Importância do evento.
- **Mensagem** - Descrição do evento.

Para esta implementação, no que toca ao elemento nível da mensagem, foram considerados 4 níveis apresentados de seguida em ordem crescente de importância:

- **DEBUG** - Nível apenas usado no ambiente de desenvolvimento com a finalidade de registar eventos relevantes para o ato de *debug*.
- **INFO** - Nível que informa sobre a normal execução dos eventos do processo.
- **WARNING** - Nível que avisa para a ocorrência de algo anormal num evento mas que não quebra a execução do processo.

- **ERROR** - Nível mais grave que indica a ocorrência de um evento que quebra a execução do processo.

Em baixo é possível ver uma parte do ficheiro log que foi gerado dia 30 de Agosto de 2019:

Listagem 4.12: Ficheiro de Log 2019-08-30.log

```
1 2019-08-30 22:00:39,806|dataprocessor_APAS|INFO|Finished processing the
    ↪ data of source FrBarrei from provider APAS
2 2019-08-30 22:00:39,808|dataprocessor_APAS|WARNING|No file for APAS
    ↪ station FrEricei in the day 2019-07-30
3 2019-08-30 22:00:39,815|dataprocessor_APAS|INFO|Processing source FrTurcif
    ↪  from provider APAS
4 2019-08-30 22:00:40,001|dataprocessor_APAS|INFO|Finished processing the
    ↪ data of source FrTurcif from provider APAS
5 2019-08-30 22:00:40,001|dataprocessor_APAS|INFO|Ready to insert data from
    ↪ provider APAS
6 2019-08-30 22:00:40,519|fetcher_pessl|INFO|Finished data retrieval from
    ↪ station 00000605
7 2019-08-30 22:00:40,521|fetcher_pessl|INFO|Starting to retrieve data from
    ↪ station 00000B43
```

4.5.5 Adição de novos Fornecedores e Fontes

A maneira como a integração dos dados meteorológicos foi implementada permite que a adição de novos fornecedores e de novas fontes de dados se faça com o mínimo de esforço possível.

Para a adição de novos fornecedores e suas fontes é necessário:

1. Criar o fornecedor, as suas fontes de dados e as séries temporais no sistema de informação.
2. Criar os ficheiros de configuração para cada uma das fontes.
3. Implementar o Fetcher para esse fornecedor (a atividade mais demorada).
4. Adicionar uma função no processo Master que invoque o Fetcher desse fornecedor e uma instância do Data Processor (como visto na listagem 4.9).

No caso da adição de novas fontes de dados de fornecedores já existentes, é necessário simplesmente:

1. Criar as fontes novas e as suas séries temporais.
2. Criar os ficheiros de configuração para essas novas fontes.

4.6 Integração dos Dados de Campo

Os dados de campo são recolhidos utilizando folhas de registo criadas e desenhadas pelos técnicos de campo com o auxílio da equipa da FCT/UNL. O desenho de cada uma das folhas foi realizado tendo em conta as necessidades dos protocolos a serem executados e as questões de logística relativas a uma visita.

As folhas de registo encontram-se criadas na Google Drive do projeto, onde toda a informação relativa a este é disponibilizada para os parceiros, através de folhas de cálculo Google Sheets. Atualmente existem 3 tipos de folhas de registo - generalista, filoxera e auxiliares - algumas delas com mais de uma versão. Quando se deslocam ao campo para fazer uma visita, os técnicos levam as folhas de registo necessárias impressas em papel onde registam toda a recolha. Depois de realizarem todas as visitas de um dia, carregam os dados manualmente para as folhas de cálculo da Google Drive. Na figura 4.6 está ilustrada a folha de registo generalista.

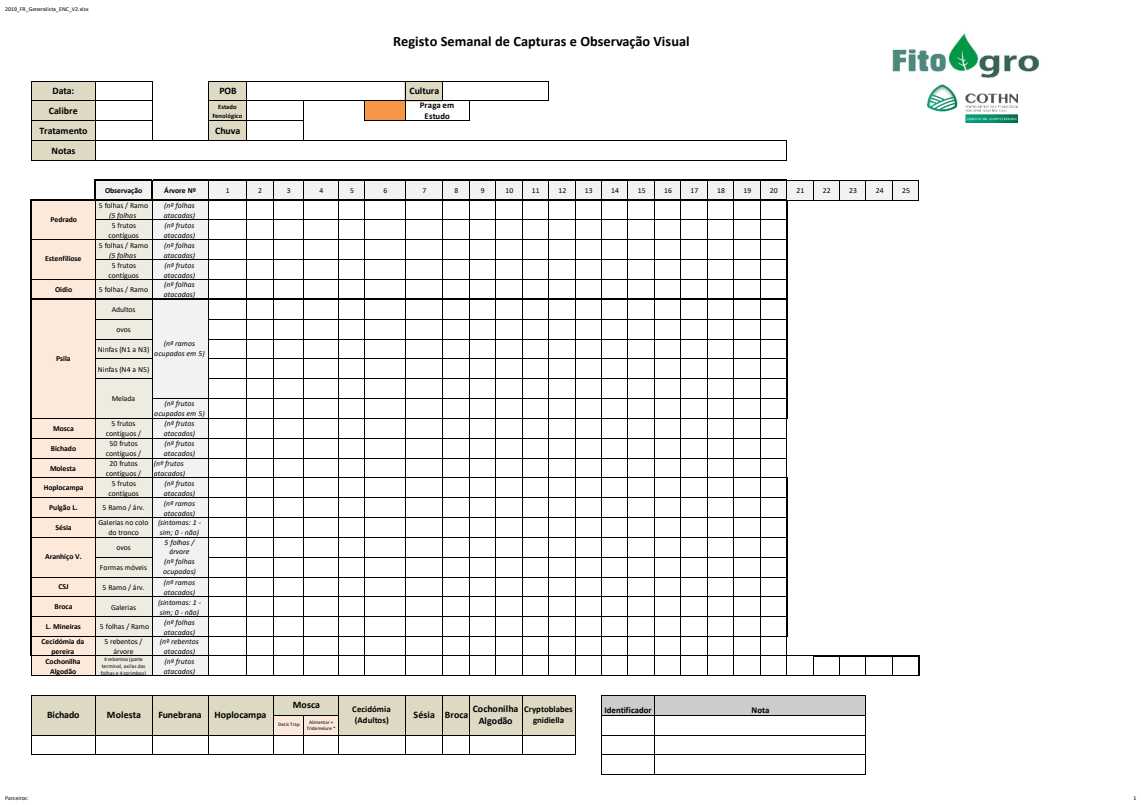


Figura 4.6: Folha de registo generalista

Um detalhe importante para a integração destes dados passou pela definição de uma estrutura para os nomes das folhas de registo. Este detalhe terá imensa importância para este processo, dado permitir declarar a partir do nome do ficheiro qual o ano em que as observações foram feitas, a parcela a que diz respeito, o tipo de folha de registo e a sua versão. Concretizando, um exemplo de um nome de folha de registo é 2019_FR_Generalista_ENC_V2 que se traduz para a folha de registo generalista versão 2 utilizada na parcela Encarnação

no ano de 2019.

Nas secções seguintes será abordada a implementação do *script* de aquisição das folhas de cálculo (4.6.1) da Google Drive passando depois para o *script* que trata da parte de extração dos dados dessas folhas e a sua inserção na base de dados (4.6.2). De notar que para a integração dos dados de campo também existe um processo master que trata de executar o *script* de aquisição e o *script* que extração e inserção sequencialmente. Na máquina em produção foi também criado um cronjob que executa este processo master semanalmente. A sua execução é realizada semanalmente devido às visitas também o serem. Quando executado são então inseridos no sistema de informação todos os dados recolhidos nas visitas dessa semana.

Devido às folhas de campo serem completamente dependentes do projeto que se estiver a tratar, toda a implementação da integração dos dados de campo foi feita *ad hoc* para o projeto FitoAgro, não sendo possível reutilizar para projetos subsequentes.

4.6.1 Aquisição

A aquisição das folhas de registo foi implementada servindo-se da Google Drive API. Para a utilizar foi inicialmente necessário pedir as credenciais de acesso a partir de uma conta Gmail que tivesse acesso à Drive do projeto. Foi posteriormente escrito um *script* parametrizável que se serve desta API para adquirir as folhas de registo. Tem como parâmetros o identificador da pasta da Google Drive onde se encontram as folhas de registo e o ano das folhas que se quer obter (este último opcional).

A validação da estrutura do nome das folhas de registo é efetuada recorrendo a uma expressão regular, sendo que só é realizado o download das folhas de registo que a cumpram. A expressão regular é a seguinte:

```
^\d{4}+_[^_\s,]+_[^_\s,]+_[^_\s,]+_[^_\s,]+$
```

Após esta validação de estrutura, são guardadas as folhas de registo que passaram, separadas em pastas por anos. A figura 4.7 ilustra o diagrama de atividades deste *script* de aquisição de folhas de registo.

4.6.2 Extração e Inserção

No que toca à extração e inserção desenvolveu-se um *script* que contém:

- Uma função por cada versão de cada folha de registo que trata da extração dos dados (utilizando a biblioteca xlrd do Python) para um ficheiro CSV comum.

- Uma função main que recebe o ano do qual se quer integrar as folhas de registo e que itera por estas, servindo-se do nome de cada uma para saber a qual função as deve entregar. Uma vez terminada esta iteração, insere os dados presentes no CSV comum na base de dados.

É importante perceber que os dados de campo estão a ser inseridos por um agente humano numa folha de cálculo da Google Drive. Este facto abre a porta para a possibilidade de ocorrência de imensos erros no *input* dos dados devido a não haver qualquer tipo de validação do que se está a fazer. Essa validação foi então implementada neste *script* de forma a reduzir ao máximo a integração de dados errados no sistema e engloba:

- Validação da estrutura da folha de registo - Verifica se a estrutura encontrada corresponde à estrutura esperada.
- Validação das datas de visita - Verifica o formato da data e se a data de uma nova visita é após a última.

Caso alguma dessas validações falhe, é reportado no log deste processo e a sua execução terminada.

Além destas validações, também são feitas algumas transformações nos valores presentes nas folhas de registo. Um exemplo de uma transformação é o seguinte: nas folhas de registo generalista de qualquer versão, quando uma célula da folha de cálculo está vazia é o mesmo que estar '0'. Visto este ser o valor mais comum, os técnicos optam por não o registar na folha devido ao trabalho excessivo de preencher cerca de 480 células com zeros.

Na figura 4.8 é possível ver o diagrama de atividades deste *script* que trata da extração e inserção destes dados.

4.7 Data Warehouse

De seguida, será abordado como foi implementado o povoamento das dimensões e das tabelas de factos de ambos os modelos multidimensionais modelados em 3.4 e o processo de publicação dos dados para a realização de visualizações.

4.7.1 Povoamento de Dimensões

Todas as dimensões (exceto as dimensões Date e Time) correspondem ao resultado de *joins* entre várias tabelas da base de dados operacional. Como tal, o problema de povoamento destas torna-se significativamente mais simples, não requerendo um processo ETL complexo. Para tratar desta questão optou-se por criar as dimensões através de vistas materializadas - tendo sempre em atenção a necessidade de excluir os registos que tenham sido *soft-deleted* - sendo apenas necessário refrescar estas para se fazer o povoamento.

As dimensões Date e Time são as únicas que têm tabela própria. O povoamento de ambas as tabelas foi realizado através de um *script* SQL, executado uma única vez, que inseriu na tabela Time todos os timestamps de um dia (segundo a segundo) e na tabela Date todas as datas entre 2010-01-01 até 2049-12-31.

Para facilitar posteriormente o mapeamento de *timestamps* para os identificadores correspondentes das tabelas Date e Time, a chave-primária destas tabelas correspondende à concatenação do ano, mês e dia para a Date (exemplo: 2019-09-06 terá como chave

primária 20190906) e hora, minuto, segundo para o Time (exemplo: 11:40:00 terá como chave primária 114000). Em linha com o que foi referido anteriormente, criou-se uma função que recebendo uma qualquer estampilha temporal calcula o identificador correspondente na dimensão Date e outra que faz o mesmo mas para a dimensão Time. Na listagem 4.13 é possível ver a função para a dimensão Date.

Listagem 4.13: Função para mapeamento de identificadores da dimensão Date

```
1 CREATE OR REPLACE FUNCTION datawarehouse.get_date_primary_key(ts timestamp
    ↪ ) RETURNS integer AS $$
2 BEGIN
3   RETURN 10000 * EXTRACT(YEAR FROM ts) +
4     100 * EXTRACT(MONTH FROM ts) +
5     EXTRACT(DAY FROM ts);
6 END;
7 $$ LANGUAGE plpgsql;
```

4.7.2 Povoamento de Factos

Pelo mesmo motivo das dimensões, as tabelas de factos foram criadas recorrendo a vistas materializadas. A listagem 4.14 mostra o comando para criação da *view* materializada correspondente à tabela de factos das séries temporais.

Listagem 4.14: Criação da tabela de factos das séries temporais

```
1  CREATE MATERIALIZED VIEW datawarehouse.seriesdata
2    AS
3    SELECT s.series_id, sd.seriesdata_seq, s.seriestype_id,
4           s.seriesvariable_id, s.datasource_id,
5           datawarehouse.get_date_primary_key(sd.seriesdata_timestamp) as
              ↪ date_id,
6           datawarehouse.get_time_primary_key(sd.seriesdata_timestamp) as
              ↪ time_id,
7           s.series_location, s.series_properties,
8           sd.seriesdata_value, sd.seriesdata_realvalue
9           sd.seriesdata_location, sd.seriesdata_properties
10   FROM series s INNER JOIN seriesdata sd on s.series_id = sd.series_id
11   WHERE s.deleted_at IS NOT NULL AND sd.deleted_at IS NOT NULL
12 WITH NO DATA;
```

Relativamente ao refrescamento do conteúdo das vistas materializadas que correspondem às dimensões e às tabelas de facto, foi implementado um *shell script* para cada um dos modelos multidimensionais com essa finalidade. Cada um desses *scripts* é executado

com a mesma frequência que é realizada a inserção desses dados no sistema - diariamente no que toca às séries temporais e semanalmente no que toca aos dados de campo.

Listagem 4.15: *Shell script* para refrescamento das dimensões e tabela de facto relativas às séries temporais

```
1 #!/bin/sh
2 psql -U fitoagro -d fitoagro -c 'REFRESH␣MATERIALIZED␣VIEW␣datawarehouse.
    ↪ seriesvariables;'
3 psql -U fitoagro -d fitoagro -c 'REFRESH␣MATERIALIZED␣VIEW␣datawarehouse.
    ↪ datasources;'
4 psql -U fitoagro -d fitoagro -c 'REFRESH␣MATERIALIZED␣VIEW␣datawarehouse.
    ↪ seriestypes;'
5 psql -U fitoagro -d fitoagro -c 'REFRESH␣MATERIALIZED␣VIEW␣datawarehouse.
    ↪ seriesdata;'
```

4.7.3 Publicação

Uma vez a componente de Data Warehouse desenvolvida foi necessário arranjar mecanismos de publicação dos dados para que se possam fazer visualizações sobre os mesmos.

Sempre que considerado necessário os conteúdos de cada dimensão e de cada tabela de factos são exportados em formato CSV. Estes ficheiros CSV são depois entregues ao Tableau Prep onde se fazem: *joins* entre as dimensões e a tabela de factos, renomeações das colunas para nomes mais indicados para apresentar em visualizações e criação de campos calculados considerados necessários (exemplo: *parsing* dos atributos JSON). Para esse efeito, criaram-se dois *flows* do Tableau Prep - um para cada data mart. O resultado final de cada um dos *flows* é uma extração que é depois entregue ao Tableau.

Optou-se por esta abordagem de extrações Tableau em vez de conexão ao vivo ao data warehouse por dois motivos principais: melhor performance e não há necessidade de acesso em tempo real ao dados.

4.8 Visualizações dos Dados

A presente secção tem como objetivo mostrar as visualizações desenvolvidas em resposta aos requisitos analíticos explicitados em 3.4.1 e explicar o seu racional. Todas as visualizações têm um exemplo ilustrativo no apêndice C. As visualizações foram as seguintes:

- **Mapa Geral** - Dá uma noção da localização geográfica de cada uma das parcelas agrícolas presentes no sistema de informação (C.1).

- **Elementos de Interesse de Parcela** - Faz zoom à localização de uma parcela. Permite ver a distribuição dos diversos elementos de interesse (C.2).

- **Parcelas em Observação, Protocolos em Execução e Número Total de Visitas** - Dashboard que permite que se tenha uma rápida *overview* dos protocolos, inimigos e parcelas em observação bem como o número total de visitas, tudo isto agrupado por campanha frutícola (C.3).

- **Tipo de Observação e Tipos de Dados por Protocolo** - Permite a visualização de todos os protocolos bem como os tipos de observação que estão a si associados e os tipos de dados destes. Visualização interessante para confirmar a correta definição dos protocolos (C.4).

- **Análise dos Valores Recolhidos** - Para cada observação de cada protocolo exibe os valores distintos que foram recolhidos (C.5). Esta visualização é particularmente interessante por permitir detetar valores fora do normal quando comparamos o tipo de dados com os valores registados.

- **Visitas por Protocolos x Parcelas** - Mostra o número de vezes que cada protocolo foi executado em cada parcela, filtrado por campanha frutícola (C.6).

- **Visitas nas Parcelas** - Para cada campanha frutícola indica para todas as parcelas em que data foram realizadas visitas. Esta visualização possibilita a deteção de quando ocorreram mais ou menos visitas que o esperado (C.7).

- **Detalhes de uma Visita** - Dashboard que mostra todos os dados recolhidos de uma visita e um mapa com localização da sua recolha caso exista (C.8).

- **Status das Fontes de Dados** - Visualização que indica de forma rápida o estado de cada uma das fontes de dados, agrupadas por fornecedor e tipo de fonte. Os diferentes estados variam consoante a data da última receção de dados, sendo estes: ativa se o sistema recebeu dados nos últimos 2 dias, atrasada se não recebeu dados nos últimos 2 dias e offline se não recebeu dados nos últimos 7 dias (C.9).

4.9 Documentação

Um dos aspetos mais importantes do desenvolvimento de software é a documentação. Uma documentação adequada contribui diretamente para a manutenção do software bem como para a redução da curva de aprendizagem de novos utilizadores. Como tal, foi criada documentação tanto para o modelo de dados como para a API REST, descritas de seguida.

4.9.1 Dicionário de Dados

Um dicionário de dados foi criado para servir de repositório central onde estão documentados todos os detalhes referentes ao modelo de dados nomeadamente: descrição de tabelas, descrição de atributos, relações entre tabelas, diagramas das várias secções, metadados e

tabelas de valor (apesar de ser um modelo genérico, as tabelas de valor foram concretizadas com exemplos do projeto FitoAgro).

Para o efeito, foi utilizado o software Dataedo [10] que foi escolhido pelas seguintes vantagens:

- Importação direta do esquema e descrições a partir do sistema de base de dados.
- Capacidade de exportar a documentação para os mais diversos formatos: HTML (navegável), PDF e Excel.
- Acesso grátis à ferramenta via a licença de Educação.

4.9.2 Documentação da API REST

No que toca à API REST, a documentação foi gerada em 3 passos:

- Criação dos *endpoints* da API REST, bem como exemplos da utilização de cada um no software Postman.
- Exportar a coleção Postman criada.
- Entregar a coleção exportada ao postmanerator.

Uma coleção Postman corresponde a um ficheiro JSON com a informação relativa aos *endpoints* e aos exemplos criados. O postmanerator é um gerador de documentação que cria automaticamente a partir dessa coleção um ficheiro HTML completamente navegável com toda a informação acerca da API.

4.10 Conclusões

A implementação do sistema de informação para se aplicar ao projeto FitoAgro foi bem sucedida. Atualmente este encontra-se em produção, cumprindo as necessidades operacionais e analíticas do projeto à data.

As tecnologias escolhidas revelaram-se adequadas, não havendo nada a apontar. O único inconveniente prende-se com o facto do Tableau necessitar de licença comercial assim que o projeto terminar.

No que respeita à parte operacional do sistema, a utilização de um servidor NGINX enquanto *proxy* reverso antes do servidor Node que aloja a API revelou-se importante para permitir o uso de HTTPS para as comunicações e também para permitir escalabilidade horizontal. Quanto ao servidor Node, a escolha do *framework* Express.js revelou-se positiva visto que permitiu que este ficasse simples e fácil de estender. De referir também que o uso do ORM Sequelize facilitou o desenvolvimento da API REST na medida em que simplificou a implementação das interações desta com a base de dados.

Relativamente à integração de dados meteorológicos, o resultado final foram dois módulos que permitem que a extensão a novos fornecedores ou a adição de novas fontes de

dados se faça com pouco esforço. A integração dos dados de campo foi realizada de uma forma completamente específica para o projeto FitoAgro.

No que toca à parte analítica do sistema, as dimensões e as tabelas de factos do Data Warehouse foram maioritariamente implementadas recorrendo a vistas materializadas algo que facilitou imenso o processo de povoamento deste componente. Foram também desenvolvidas um conjunto de visualizações que, bebendo os dados publicados a partir do Data Warehouse, permitiram responder aos requisitos analíticos elicitados para o projeto FitoAgro.

Por fim, foi criada documentação adequada para o modelo de dados e para a API REST de forma a dar suporte tanto aos utilizadores que utilizem a API como os programadores que no futuro venham a manter/estender a solução implementada.

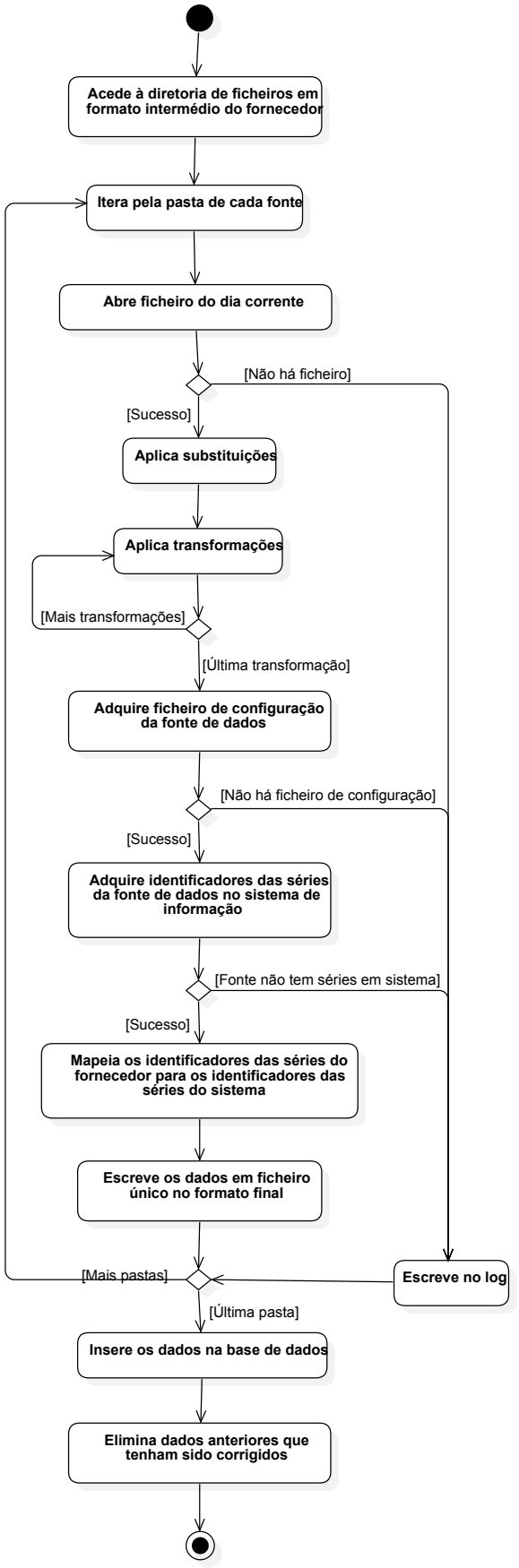


Figura 4.5: Diagrama de atividades do Data Processor

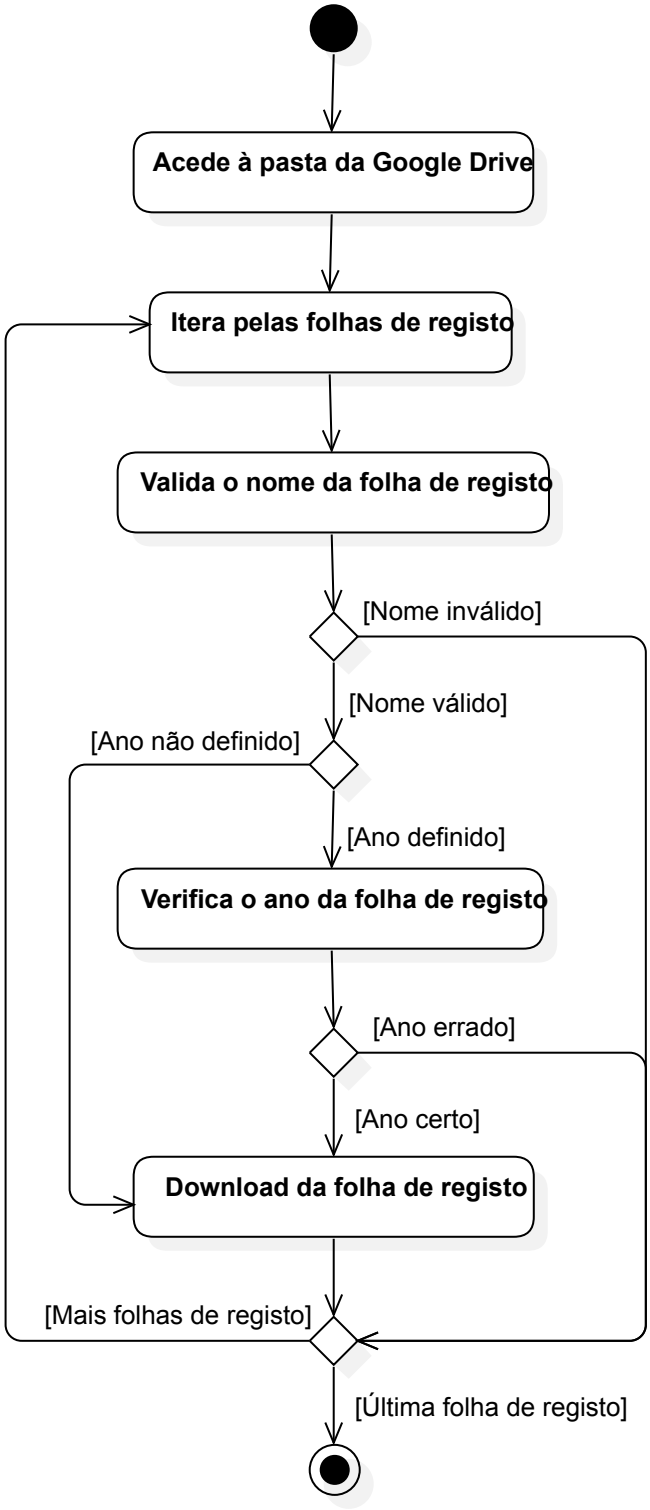


Figura 4.7: Diagrama de atividades do processo de aquisição das folhas de registo

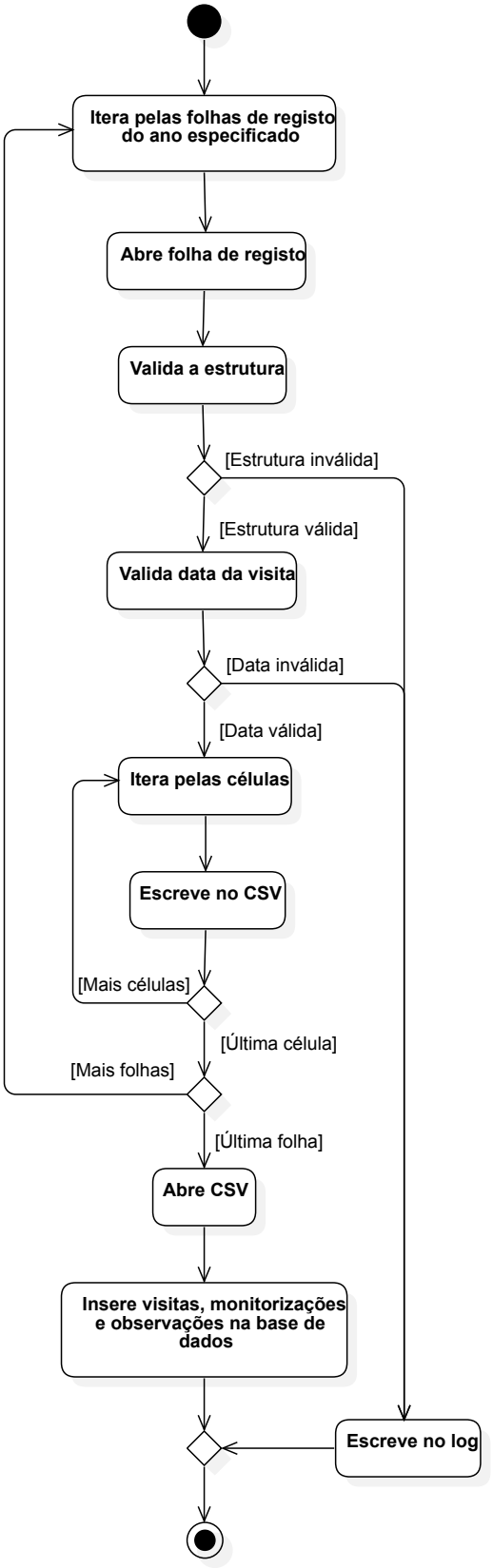


Figura 4.8: Diagrama de atividades do processo de extração e inserção dos dados das folhas de registo

Capítulo 5

Avaliação

Neste capítulo vai-se proceder à avaliação do resultado final desta dissertação recorrendo a alguns aspetos considerados mais relevantes: a aplicabilidade do sistema de informação a projetos do mesmo domínio que o FitoAgro, a usabilidade da API desenvolvida e por fim o desempenho do sistema recorrendo a alguns testes realizados com essa finalidade.

5.1 Aplicabilidade a projetos do mesmo domínio

Uma das questões mais importantes que se pode levantar relativamente ao sistema de informação que se implementou no âmbito desta dissertação é a seguinte: o sistema aplica-se a projetos do mesmo domínio que o FitoAgro?

Para responder à mesma recorrer-se-à a um projeto já referido neste documento, o SafeBrocolo. O projeto SafeBrocolo possuía requisitos iguais ao FitoAgro - de forma resumida, este precisava de um sistema de informação que permitisse armazenar e disponibilizar os dados recolhidos em campo e os dados meteorológicos para que se pudesse realizar o estudo de problemas que tinham vindo a afetar as culturas de brócolos. Tendo em conta esta informação, uma maneira talvez um pouco ingénua de responder à questão levantada nesta secção seria concluir que se o projeto FitoAgro tem requisitos semelhantes ao projeto SafeBrocolo e se o sistema desenvolvido nesta dissertação colmata as necessidades do FitoAgro então seria aplicável também ao SafeBrocolo.

Contudo, este argumento facilmente levanta a questão contrária: o sistema desenvolvido para o SafeBrocolo não é aplicável ao FitoAgro porquê? Ora, a resposta é negativa devido a um facto essencial: o sistema do SafeBrocolo foi uma solução desenhada à medida desse projeto. No caso do sistema apresentado neste documento, o seu desenho não foi orientado a um projeto específico mas sim ao domínio em questão - projetos de investigação da área agrícola, focados no estudo, prevenção e controlo de pragas. Isto levou a uma preocupação

de se atingir um certo nível de abstração quando se modelou o sistema, resultando na abordagem genérica demonstrada no capítulo 3 e tendo depois a sua utilidade demonstrada através da implementação da instância que serve o FitoAgro. Adicionalmente, a solução apresentada neste documento já contempla algumas das sugestões de trabalho futuro do SafeBrocolo que são relevantes para este tipo de projetos: representação de secções de parcelas, informação acerca de recolhas de amostras e informações acerca de eventos que ocorram nas parcelas.

Para concluir o raciocínio, resta então explicar os passos que seriam necessários para aplicar este sistema genérico ao SafeBrocolo:

- Criação da base de dados e *deployment* do servidor - Inicialmente ter-se-ia de criar a base de dados, configurando o esquema JSON dos atributos Properties e Metadata de cada tabela à medida das necessidades específicas que houver. *Deployment* do servidor para a API REST ficar disponível.
- Atualização da aplicação móvel - Este projeto contou com o desenvolvimento de uma aplicação móvel para recolha dos dados de campo. Esta precisaria de ter atualizadas todas as chamadas que faz ao servidor de acordo com a API REST exposta pelo servidor.
- Definição dos processos de integração de dados meteorológicos - Os fornecedores, as fontes de dados e as séries destas a serem integradas no sistema teriam de ser criadas. Posteriormente, criar-se-ia um processo Master, tal como se fez para o projeto FitoAgro, que utiliza o módulo Fetcher e o módulo Data Processor. Para o caso de ter fornecedores cujo Fetcher já tenha sido implementado, é reutilizá-los. Caso hajam fornecedores cujos Fetchers ainda não existam, é necessário implementá-los.

Deste modo fica provado que sim, o sistema genérico desenvolvido é aplicável a projetos do mesmo domínio que o FitoAgro.

5.2 Testes de Desempenho

Ao longo desta secção será dado conhecimento de alguns testes que foram realizados de forma a testar o desempenho do sistema de informação em alguns cenários.

5.2.1 Características do Sistema

Todos os testes de desempenho foram realizados na máquina de desenvolvimento, Lenovo ThinkPad T480, cujas características são:

- Processador: Intel Core i7-8650U @ 1.90GHz
- Memória: 32 GB 2400 MHz DDR4
- GPU: Intel UHD Graphics 620

- Disco: 512 GB SSD, com cerca de 385 GB livres
- Sistema Operativo: Windows 10 Pro versão 1803

5.2.2 Integração dos Dados Meteorológicos

Tal como foi referido anteriormente, o processo de integração dos dados meteorológicos é executado diariamente, inserindo no sistema os dados de 59 fontes de dados diferentes. De notar que o número de séries temporais geradas por cada fonte varia (entre 7 a 15 séries) bem como o período em que estas são medidas (de 15 em 15 minutos, de 30 em 30 minutos ou de hora a hora).

A execução deste processo no dia 6 de Setembro de 2019 resultou na inserção de 55138 valores, com um tempo de execução de 51,516 segundos. A tabela 5.1 apresenta o tempo de execução (em segundos) de cada uma das *threads* deste processo, detalhado ao nível de cada um dos seus sub-processos (Fetch, Process, Integrate).

Thread	**Fetch**	**Process**	**Integrate**	**Total**
APAS	17,681	0,828	12,684	31,193
IPMA (estações)	3,723	0,234	12,123	16,080
IPMA (previsões)	4.012	0,747	23.415	28,174
PESSL/FieldClimate	25,804	0,718	24,994	51,516
TerraPro	-	-	-	-
WiseCrop	23,101	0,235	20,964	44,300

Tabela 5.1: Tempo de execução por *thread*

Nota: O fornecedor TerraPro não apresenta tempo de execução uma vez que não entregou dados nesse dia. As durações foram extraídas a partir do log desse dia.

5.2.3 Integração dos Dados de Campo

Relativamente ao processo de integração dos dados de campo, este é executado semanalmente aos domingos de forma a inserir todos os dados recolhidos durante essa semana. No ano de 2019 o projeto FitoAgro conta com 13 parcelas agrícolas em observação e os técnicos fazem uso de 17 folhas de registo. Isto resulta em cerca de 6000 novas observações por semana.

A execução deste processo no dia 8 de Setembro de 2019 resultou na inserção de 92 visitas, 1533 monitorizações e 44564 observações, recolhidas entre o dia 14 de Julho de 2019 e o dia da execução. Os processos tinham vindo a falhar semanalmente devido a uma desformatação numa das folhas de registo que ainda não tinha sido resolvida pelos técnicos de campo.

O processo no dia referido demorou 182,109 segundos repartidos da seguinte forma:

- Aquisição: 59,694 segundos
- Extração: 27,559 segundos

- Inserção: 94,856 segundos

Nota: As durações foram extraídas a partir do log desse dia.

5.2.4 Consultas aos Dados

O último teste realizado incidiu no desempenho de consultas realizadas aos dados meteorológicos. Para esse efeito realizou-se uma consulta que pedia os dados todos da estação COTHN do fornecedor PESSL/fieldclimate desde o dia 1 de Janeiro de 2017 até ao dia 6 de Setembro de 2019.

A consulta, executada diretamente na base de dados, demorou 172 milissegundos e devolveu 535863 tuplos. A criação do índice na tabela Series Data contribuiu diretamente para a rápida execução desta consulta.

Adicionalmente, para se perceber se haviam diferenças significativas, a mesma consulta foi também realizada através de um pedido para a API REST através do *endpoint* correspondente nos dois formatos disponibilizados - JSON e Excel. Foram devolvidos os mesmos 535863 resultados tendo demorado 21.848 segundos no caso do Excel e 15,968 segundos no caso de JSON já contabilizando o tempo de download.

5.3 Conclusões

O sistema de informação desenvolvido consegue ser aplicado sem grande esforço a projetos do mesmo domínio que o projeto FitoAgro. Isto é resultado da modelação genérica e implementação realizada.

No que toca ao desempenho do sistema, é preciso ter em consideração que os testes foram realizados num computador portátil com memória e capacidade de processamento limitada. Não obstante, os resultados foram satisfatórios.

A integração dos dados meteorológicos mostrou-se bastante eficiente, demorando em média menos de um segundo por fonte de dados. O uso de *threading* neste processo foi essencial para maximizar a sua performance.

Passando para o processo de integração dos dados de campo (implementado à medida do FitoAgro), este apresentou resultados razoáveis. O sub-processo de inserção foi aquele cuja execução foi mais demorada, algo que pode ser justificado devido a este fazer inserções em 3 tabelas sequencialmente (Visits, Monitorizations e Observations).

Nas consultas aos dados meteorológicos, a consulta via API mostrou-se, como esperado, menos eficiente que os resultados a consultas realizadas diretamente à base de dados. Isto deve-se a transformações de *output* que são realizadas ao nível da API.

Com base no que foi apresentado, é então possível concluir que o sistema desenvolvido pode ser reutilizado para projetos da área sem grande dificuldade, não apresentando qualquer problema de desempenho.

Capítulo 6

Conclusões e Trabalho Futuro

Este último capítulo sumariza todo o trabalho realizado ao longo desta dissertação bem como os resultados que foram alcançados. Após este sumário em jeito de conclusão, termina-se o documento com um conjunto de sugestões de trabalho futuro.

6.1 Conclusões

Esta dissertação assumiu como objetivo a conceção de um sistema de informação genérico para projetos de índole fitossanitária bem como a sua aplicação a um caso concreto, o projeto FitoAgro.

O resultado final pode ser considerado um sucesso pois o sistema encontra-se funcional e a servir adequadamente as necessidades desse projeto.

O modelo de dados concebido apresenta-se genérico, tendo não só a capacidade de representar todo o tipo de informação necessária a estes projetos mas também a valência de ser extensível e flexível às necessidades específicas de cada um.

O Data Warehouse implementado teve como base dois modelos multidimensionais que resultaram de uma abordagem geral mas que cumpre os requisitos analíticos do FitoAgro. Projetos futuros que tenham necessidades analíticas poderão fazer uso deste, caso contrário deverá ser estendido com novos modelos multidimensionais.

Do mesmo modo, as visualizações implementadas tiveram como base as necessidades analíticas do FitoAgro, podendo ser reutilizadas caso se considerem adequadas. De outra forma, requererá sempre a implementação de novas visualizações mais adequadas. O Tableau poderá também não ser a tecnologia mais indicada para uso contínuo no sistema caso existam restrições financeiras, pois necessita de licença assim que o projeto em questão acabe.

O servidor manifesta capacidade de escalar e de ser estendido através da arquitetura

deste e das tecnologias utilizadas, pelo que não deverá ter qualquer problema em contemplar projetos de dimensão superior ao FitoAgro ou com a adição de novos componentes. A API REST exposta por este é simples de utilizar, está bem documentada e permite o acesso a toda a informação, podendo ainda ver a sua usabilidade melhorada caso se dê o upgrade do nível 2 para nível 3 do Modelo de Maturidade Richardson.

A integração dos dados meteorológicos foi implementada de forma genérica e apresentou um bom desempenho, podendo ser reutilizada em projetos futuros. Por outro lado, a integração dos dados de campo foi realizada *ad hoc* para o FitoAgro pelo que deverá ser reimplementada para projetos futuros. A integração deste tipo de dados revelou diversos problemas ao longo do tempo como erros de *inputs*, erro nos nomes atribuídos aos ficheiros aquando a sua criação e desformatação não intencional da folha de cálculo na altura da inserção dos dados. No futuro deverá procurar-se uma solução melhor para esta problemática.

6.2 Trabalho Futuro

Apesar de todo o trabalho realizado cumprir os objetivos da dissertação, existem alguns aspectos que podem/devem vir a ser implementados em versões futuras com o objetivo de acrescentar valor ao sistema de informação genérico desenvolvido e à sua instância implementada para o projeto FitoAgro.

6.2.1 Incorporação de Dados de Satélite

Os satélites são uma fonte de dados que podem ser utilizados para complementar os dados meteorológicos que o sistema de informação já contempla. Desse modo, o módulo de Séries Temporais do modelo de dados poderá ser estendido no futuro de forma a suportar rasters.

6.2.2 Implementação da Camada Cliente

Como foi referido, a Camada Cliente não foi implementada no âmbito desta dissertação. No entanto, a implementação de dois componentes desta camada seriam sem dúvida uma mais valia, sendo eles uma aplicação móvel genérica (tal como o sistema de informação) para recolha dos dados de campo e uma aplicação web com o objetivo de fazer a gestão dos dados (exemplo: criar parcelas, criar protocolos, etc.). A aplicação móvel facilitaria a recolha dos dados de campo tanto em termos de logística das visitas (os técnicos não precisariam de andar com folhas de papel) como na qualidade dos dados visto que reduziria ao máximo os erros de *input* do utilizador e acabaria com os erros relativos à integração destes no sistema. A aplicação web para além da tal gestão ainda poderia permitir a migração da Camada de Visualizações já que o Tableau necessitará de aquisição de licença assim que o projeto FitoAgro acabar.

6.2.3 HATEOAS na API REST

Uma das melhorias possíveis para a API REST seria passá-la para o nível 3 do Modelo de Maturidade Richardson. O nível 3 deste modelo consiste em tudo o que o nível 2 tem mais a adição do chamado HATEOAS (Hypermedia As The Engine Of Application State). A ideia por detrás do HATEOAS é que um cliente após mandar um pedido para a API consegue dinamicamente descobrir as ações disponíveis e aceder a todos os recursos que precisa através de *hypermedia links* devolvidos na resposta. Esta melhoria seria uma extensão da API já implementada que traria vantagens como facilidade de uso e de descoberta para os futuros utilizadores desta.

6.2.4 Alerta para erros na Integração dos Dados de Campo

Quando ocorrem erros nos processos de integração dos dados de campo, estes ficam simplesmente registados no log. Isto obriga a que periodicamente alguém verifique se não ocorreu nada de inesperado na execução deste processo. Uma solução a implementar que facilitaria este processo seria, em caso de ocorrência de erros, enviar um mail a alguém responsável com o log. Isto agilizaria o processo de correção dos erros, principalmente se tiverem sido consequência de problemas de *input* nas folhas de registo por parte dos técnicos de campo.

6.2.5 Novas Visualizações

Como foi referido no documento, ao dia da escrita deste documento não tinham sido levantados os requisitos analíticos por parte dos cientistas do projeto FitoAgro. Como tal, as visualizações que foram implementadas no Tableau são estritamente relativas ao controlo da qualidade dos dados.

Assim que sejam levantados e comunicados deverão ser implementadas visualizações que os satisfaçam.

6.2.6 Modelos Preditivos

Uma extensão natural ao trabalho realizado será a implementação de modelos preditivos. Falando especificamente do projeto FitoAgro, o suporte dado à equipa científica pelo sistema de informação resultará na definição deste tipo de modelos para as pragas emergentes do Oeste. Esses modelos deverão depois ser implementados de forma a que possam ser validados. Desta forma, um trabalho futuro a realizar será o desenvolvimento de um componente genérico que permita a execução de modelos preditivos, guardando os seus resultados em sistema, estando já incorporado no modelo uma entidade genérica para o efeito.

Bibliografia

[1] *Accumulo*. URL: https://accumulo.apache.org/.

[2] *Accuweather API*. URL: https://developer.accuweather.com/packages.

[3] S. Agersten e H. Heiberg. *Naming convention for climate and observation data*. Rel. téc. 13. METNorway, set. de 2017.

[4] L. Alarabi e M. F. Mokbel. *A Demonstration of ST-Hadoop: A MapReduce Framework for Big Spatio-temporal Data*. 2017. URL: www.vldb.org/pvldb/vol10/p1961-alarabi.pdf.

[5] D. V. P. Canário. *A problemática das cochonilhas-algodão em cultura protegida de hortícolas na região Oeste*. 2016. URL: http://hdl.handle.net/10400.5/12113.

[6] *Cecidómia*. URL: https://infoagro.cothn.pt/portal/index.php?id=1307.

[7] *CF Standard Names*. URL: http://cfconventions.org/standard-names.html.

[8] C. Coutinho. *O bichado (Cydia pomonella L.) em pomóideas*. First. Núcleo de Documentação e Relações Públicas, 2011. ISBN: 978-989-8201-30-0. URL: http://www.drapn.min-agricultura.pt/drapn/conteudos/ft2010/ficha_tecnica_37_2011.pdf.

[9] *Cropio*. URL: https://cropio.com/.

[10] *Dataedo*. URL: https://dataedo.com/.

[11] *Django*. URL: https://www.djangoproject.com/.

[12] M. Drusch, U. D. Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, A. Meygret, F. Spoto, O. Sy, F. Marchese e P. Bargellini. "Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services". Em: *Remote Sensing of Environment* 120 (2012). The Sentinel Missions - New Opportunities for Science, pp. 25 –36. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2011.11.026. URL: http://www.sciencedirect.com/science/article/pii/S0034425712000636.

[13] *ecCodes Home*. URL: https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home.

[14] A. Eldawy e M. F. Mokbel. "Pigeon: A spatial MapReduce language". Em: *IEEE 30th International Conference on Data* (2014). DOI: 10.1109/ICDE.2014.6816751.

[15] *EUMETSAT*. URL: https://www.eumetsat.int/website/home/AboutUs/WhoWeAre/index.html.

[16] *Filoxera*. URL: https://infoagro.cothn.pt/portal/index.php?id=1320.

[17] *Filoxera del peral, Aphanostigma pyri, descripción, daños y control integrado*. URL: http://www.agroes.es/cultivos-agricultura/cultivos-frutales-y-fruticultura/manzano/1314-filoxera-del-peral-aphanostigma-pyri.

[18] *FitoAgro*. URL: https://inovacao.rederural.gov.pt/grupos-operacionais/13-projectos-groupos-operacionais/87-fitoagro.

[19] *Flask web development, one drop at a time*. URL: http://flask.pocoo.org/.

[20] *Geomesa Architecture Overview*. URL: http://www.geomesa.org/documentation/user/architecture.html.

[21] T. H. Group. *HDF5 User's Guide*. URL: https://support.hdfgroup.org/HDF5/doc/UG/HDF5_Users_Guide.pdf.

[22] *Guidelines for Construction of CF Standard Names*. URL: http://cfconventions.org/Data/cf-standard-names/docs/guidelines.html.

[23] *HDF5 for Python*. URL: https://www.h5py.org/.

[24] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert e M. Ronquest. "GeoMesa: a distributed architecture for spatio-temporal fusion". Em: 9473 (2015). DOI: 10.1117/12.2177233.

[25] M. Immitzer, F. Vuolo e C. Atzberger. "First Experience with Sentinel-2 Data for Crop and Tree Species Classifications in Central Europe". Em: *Remote Sensing* 8.3 (2016). ISSN: 2072-4292. DOI: 10.3390/rs8030166. URL: http://www.mdpi.com/2072-4292/8/3/166.

[26] O. G. C. Inc. *OGC Vision, Mission, & Goals*. URL: http://www.opengeospatial.org/ogc/vision.

[27] O. G. C. Inc. *OpenGIS Web Map Server Implementation Specification*. 2006. URL: http://portal.opengeospatial.org/files/?artifact_id=14416.

[28] O. G. C. Inc. *OpenGIS Web Feature Service 2.0 Interface Standard – With Corrigendum*. 2014. URL: https://portal.opengeospatial.org/files/09-025r2.

[29] *IPMA API*. URL: http://api.ipma.pt/.

[30] J. R. Irons, J. L. Dwyer e J. A. Barsi. "The next Landsat satellite: The Landsat Data Continuity Mission". Em: *Remote Sensing of Environment* 122 (2012). Landsat Legacy Special Issue, pp. 11 –21. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2011.08.026. URL: http://www.sciencedirect.com/science/article/pii/S0034425712000363.

[31] *JAVA HDF5 INTERFACE (JHI5)*. URL: https://support.hdfgroup.org/products/java/JNI3/jhi5/.

[32] D. Johnson, R. Bessin e L. Townsend. *Predicting insect development using degree days*. URL: https://entomology.ca.uky.edu/files/efpdf1/ef123.pdf.

[33] *Land Surface Temperature (MLST)*. URL: https://landsaf.ipma.pt/en/products/land-surface-temperature/lst/.

[34] G. Leote. *Field InformationWeb Platform for Agricultural Applications*. Set. de 2013.

[35] *Let's Encrypt*. URL: https://letsencrypt.org/.

[36] J. Li. *A Review of Spatial Interpolation Methods for Environmental Scientists*. Jan. de 2008. URL: https://www.researchgate.net/publication/246546630_A_Review_of_Spatial_Interpolation_Methods_for_Environmental_Scientists.

[37] *LSA SAF*. URL: https://landsaf.ipma.pt/en/.

[38] O. C. Maloy. *Plant Disease Management*. 2005. DOI: 10.1094/PHI-I-2005-0202-01. URL: http://www.apsnet.org/edcenter/intropp/topics/Pages/PlantDiseaseManagement.aspx.

[39] A. J. A. Martins. *As cochonilhas-algodão da vinha (Hemiptera: Pseudococcidae) na região do Oeste*. 2009. URL: http://hdl.handle.net/10400.5/2053.

[40] *NetCDF Libraries*. URL: https://www.unidata.ucar.edu/downloads/netcdf/index.jsp.

[41] *netcdf4-python*. URL: http://unidata.github.io/netcdf4-python/.

[42] J. K. Nidzwetzki e R. H. Güting. "Distributed secondo: an extensible and scalable database management system". Em: *Distributed and Parallel Databases* 35.3 (dez. de 2017), pp. 197–248. ISSN: 1573-7578. DOI: 10.1007/s10619-017-7198-9. URL: https://doi.org/10.1007/s10619-017-7198-9.

[43] *Node.js*. URL: https://nodejs.org/en/.

[44] *Novo Bichado*. URL: https://infoagro.cothn.pt/portal/index.php?id=1284.

[45] *OpenWeatherMap API*. URL: https://openweathermap.org/price.

[46] W. M. ORGANIZATION. *A GUIDE TO THE CODE FORM FM-94 BUFR*. URL: https://www.wmo.int/pages/prog/www/WDM/Guides/Guide-binary-1A.html.

[47] W. M. ORGANIZATION. *Introduction to GRIB Edition1 and GRIB Edition 2*. URL: https://www.wmo.int/pages/prog/www/WMOCodes/Guides/GRIB/Introduction_GRIB1-GRIB2.pdf.

[48] F. Pinto. *Sistema para acompanhamento da cultura do brócolo*. Mar. de 2017.

[49] *PostGIS*. URL: http://postgis.net/.

[50] *PostgreSQL*. URL: https://www.postgresql.org/.

[51] *Raster / Vector Data Models*. URL: http://www.catalonia.org/cartografia/Clase_03/Raster_Vector.html.

[52] R. Rew e G. Davis. "NetCDF: an interface for scientific data access". Em: *IEEE Computer Graphics and Applications* 10.4 (jul. de 1990), pp. 76–82. ISSN: 0272-1716. DOI: 10.1109/38.56302.

[53] *Richardson Maturity Model*. URL: https://martinfowler.com/articles/richardsonMaturityMode html.

[54] *Ruby on Rails*. URL: https://rubyonrails.org/.

[55] J. M. da Silva, C. Damásio, A. Sousa, L. Bugalho, L. Pessanha e P. Quaresma. "Agriculture pest and disease maps considering MSG satellite data and land surface temperature". Em: *International Journal of Applied Earth Observation and Geoinformation* 38 (2015), pp. 40–50. DOI: 10.1016/j.jag.2014.12.016.

[56] *SpatiaLite*. URL: https://www.gaia-gis.it/fossil/libspatialite/index.

[57] *SQLite*. URL: https://www.sqlite.org/index.html.

[58] *USGS Global Visualization Viewer*. URL: https://glovis.usgs.gov/.

[59] *Web server performance comparison*. URL: https://help.dreamhost.com/hc/en-us/articles/215945987-Web-server-performance-comparison.

[60] *weeWX*. URL: http://www.weewx.com/.

[61] *WiseCrop*. URL: https://www.wisecrop.com/.

[62] Z. Xun, E. Zhong, L. Wu, S. Weiwei, S. Wang e X. Zhang. "A GDB-CLI based BUFR meteorological data engine". Em: (dez. de 2012), pp. 1121–1125.

APÊNDICE A

Mapeamento das Entidades

Nome no Diagrama	**Nome no Documento**
Agents	Agentes
Agent Types	Tipos de Agentes
Organizations	Organizações
Organization Types	Tipos de Organizações
Entities	Entidades
Organization Members	Membros de Organizações
Member Types	Tipos de Membros
Permissions	Permissões
Regions	Regiões
Region Types	Tipos de Regiões
Topological Relations	Relações Topológicas
Topological Relation Types	Tipos de Relações Topológicas
Elements of Interest	Elementos de Interesse
Element of Interest Types	Tipos de Elementos de Interesse
Crops	Culturas
Crop Types	Tipos de Culturas
Varieties	Variedades
Phenological Stages	Estados Fenológicos
Phenological Stages Durations	Durações de Estados Fenológicos
Varieties in Regions	Variedades Em Regiões
Organisms	Organismos
Organism Types	Tipos de Organismos
Prediction Models	Modelos Preditivos

Organisms Monitored	Organismos Monitorizados
Periods	Períodos
Period Types	Tipos de Períodos
Regions in Periods	Regiões em Períodos
Visits	Visitas
Monitorizations	Monitorizações
Monitorizations by Agents	Monitorizações de Agentes
Observations	Observações
Observation Types	Tipos de Observações
Protocols	Protocolos
Protocols on Regions in Periods	Protocolos em Regiões em Períodos
Observations of Protocols	Observações de Protocolos
Samples	Amostras
Sample Types	Tipos de Amostras
Events	Eventos
Event Types	Tipos de Eventos
Event Consequences	Consequências de Eventos
Media	Multimédia
Media Types	Tipos de Multimédia
Data Providers	Fornecedores de Dados
Data Provider Types	Tipos de Fornecedores de Dados
Data Sources	Fontes de Dados
Data Source Types	Tipos de Fontes de Dados
Data Source Dependences	Dependências de Fontes de Dados
Regions Main Data Sources	Fontes de Dados de Regiões
Series	Séries
Series Data	Dados das Séries
Series Types	Tipos de Séries
Series Relations	Relações de Séries
Series Relation Types	Tipos de Relações de Séries
Regions in Periods Series	Séries de Regiões em Períodos
Series Variables	Identificadores de Séries
Calculation Methods	Métodos de Cálculo
Series Periods	Períodos de Séries
Data Types	Tipos de Dados
Units	Unidades
Settings	Settings
Properties Schemas	Esquemas de Propriedades
Metadata Schemas	Esquemas de Metadados

APÊNDICE B

Listagem de Casos de Uso da API REST

- **Geral**

  C1 - O utilizador autentica-se de forma a receber o token de acesso.

  C2 - O utilizador consulta a versão corrente de API.

- **Agentes**

  C3 - O utilizador consulta os agentes.

  C4 - O utilizador cria um agente.

  C5 - O utilizador edita um agente.

  C6 - O utilizador elimina um agente.

- **Tipos de Agentes**

  C7 - O utilizador consulta os tipos de agentes.

  C8 - O utilizador cria um tipo de agente.

  C9 - O utilizador edita um tipo de agente.

  C10 - O utilizador elimina um tipo de agente.

- **Organizações**

  C11 - O utilizador consulta as organizações.

  C12 - O utilizador cria uma organização.

  C13 - O utilizador edita uma organização.

  C14 - O utilizador elimina uma organização.

- **Tipos de Organizações**

C15 - O utilizador consulta os tipos de organizações.

C16 - O utilizador cria um tipo de organização.

C17 - O utilizador edita um tipo de organização.

C18 - O utilizador elimina um tipo de organização.

- **Membros de Organizações**

  C19 - O utilizador consulta os agentes de uma organização.

  C20 - O utilizador consulta as organizações de um agente.

  C21 - O utilizador cria a associação entre um agente e uma organização.

  C22 - O utilizador edita a associação entre um agente e uma organização.

  C23 - O utilizador elimina a associação entre um agente e uma organização.

- **Tipos de Membros**

  C24 - O utilizador consulta os tipos de membros.

  C25 - O utilizador cria um tipo de membro.

  C26 - O utilizador edita um tipo de membro.

  C27 - O utilizador elimina um tipo de membro.

- **Permissões**

  C28 - O utilizador consulta as permissões de cada agente.

  C29 - O utilizador consulta as permissões para cada região.

  C30 - O utilizador cria uma permissão.

  C31 - O utilizador edita uma permissão.

  C32 - O utilizador elimina uma permissão.

- **Regiões**

  C33 - O utilizador consulta as regiões.

  C34 - O utilizador cria uma região.

  C35 - O utilizador edita uma região.

  C36 - O utilizador elimina uma região.

- **Tipos de Regiões**

  C37 - O utilizador consulta os tipos de regiões.

  C38 - O utilizador cria um tipo de região.

  C39 - O utilizador edita um tipo de região.

  C40 - O utilizador elimina um tipo de região.

- **Relações Topológicas**

  C41 - O utilizador consulta todas as relações topológicas de uma região.

  C42 - O utilizador cria uma relação topológica entre duas regiões.

  C43 - O utilizador edita uma relação topológica entre duas regiões.

  C44 - O utilizador elimina uma relação topológica entre duas regiões.

- **Tipos de Relações Topológicas**

  C45 - O utilizador consulta os tipos de relações topológicas.

  C46 - O utilizador cria um tipo de relação topológica.

  C47 - O utilizador edita um tipo de relação topológica.

  C48 - O utilizador elimina um tipo de relação topológica.

- **Elementos de Interesse**

  C49 - O utilizador consulta os elementos de interesse.

  C50 - O utilizador cria um elemento de interesse.

  C51 - O utilizador edita um elemento de interesse.

  C52 - O utilizador elimina um elemento de interesse.

- **Tipos de Elementos de Interesse**

  C53 - O utilizador consultar os tipos de elementos de interesse.

  C54 - O utilizador cria um tipo de elemento de interesse.

  C55 - O utilizador edita um tipo de elemento de interesse.

  C56 - O utilizador elimina um tipo de elemento de interesse.

- **Culturas**

  C57 - O utilizador consulta as culturas.

  C58 - O utilizador cria uma cultura.

  C59 - O utilizador edita uma cultura.

  C60 - O utilizador elimina uma cultura.

- **Tipos de Culturas**

  C61 - O utilizador consulta os tipos de culturas.

  C62 - O utilizador cria um tipo de cultura.

  C63 - O utilizador edita um tipo de cultura.

  C64 - O utilizador elimina um tipo de cultura.

- **Variedades**

  C65 - O utilizador consulta as variedades de uma cultura.

  C66 - O utilizador cria uma variedade.

  C67 - O utilizador edita uma variedade.

  C68 - O utilizador elimina uma variedade.

- **Estados Fenológicos**

  C69 - O utilizador consulta os estados fenológicos de uma cultura.

  C70 - O utilizador cria um estado fenológico.

  C71 - O utilizador edita um estado fenológico.

  C72 - O utilizador elimina um estado fenológico.

- **Durações de Estados Fenológicos**

  C73 - O utilizador consulta as durações dos estados fenológicos de uma variedade.

  C74 - O utilizado cria a duração de um estado fenológico.

  C75 - O utilizado edita a duração de um estado fenológico.

  C76 - O utilizado elimina a duração de um estado fenológico.

- **Variedades em Regiões**

  C77 - O utilizador consulta as variedades de uma região.

  C78 - O utilizador consulta as regiões de uma variedade.

  C79 - O utilizador cria a associação entre uma região e uma variedade.

  C80 - O utilizador edita a associação entre uma região e uma variedade.

  C81 - O utilizador elimina a associação entre uma região e uma variedade.

- **Inimigos**

  C82 - O utilizador consulta os inimigos.

  C83 - O utilizador cria um inimigo.

  C84 - O utilizador edita um inimigo.

  C85 - O utilizador elimina um inimigo.

- **Tipos de Inimigos**

  C86 - O utilizador consulta os tipos de inimigos.

  C87 - O utilizador cria um tipo de inimigo.

  C88 - O utilizador edita um tipo de inimigo.

  C89 - O utilizador elimina um tipo de inimigo.

- **Modelos de Inimigos**

  C90 - O utilizador consulta os modelos de inimigos.

  C91 - O utilizador cria um modelo de inimigo.

  C92 - O utilizador edita um modelo de inimigo.

  C93 - O utilizador elimina um modelo de inimigo.

- **Inimigos Monitorizados**

  C94 - O utilizador consulta os inimigos monitorizados numa região num dado período.

  C95 - O utilizador consulta as regiões num dado período em que o inimigo foi monitorizado.

  C96 - O utilizador cria a associação entre uma região em período e um inimigo.

  C97 - O utilizador edita a associação entre uma região em período e um inimigo.

  C98 - O utilizador elimina a associação entre uma região em período e um inimigo.

- **Períodos**

  C99 - O utilizador consulta os períodos.

  C100 - O utilizador cria períodos.

  C101 - O utilizador edita períodos.

  C102 - O utilizador elimina períodos.

- **Tipos de Períodos**

  C103 - O utilizador consulta os tipos de períodos.

  C104 - O utilizador cria um tipo de período.

  C105 - O utilizador edita um tipo de período.

  C106 - O utilizador elimina um tipo de período.

- **Regiões em Períodos**

  C107 - O utilizador consulta os períodos em que uma região participou.

  C108 - O utilizador consulta as regiões que participaram num período.

  C109 - O utilizador cria a associação entre uma região e um período.

  C110 - O utilizador edita a associação entre uma região e um período.

  C111 - O utilizador elimina a associação entre uma região e um período.

- **Visitas**

  C112 - O utilizador consulta as visitas realizadas numa região em período.

  C113 - O utilizador cria uma visita.

  C114 - O utilizador cria tudo o que foi recolhido numa visita (monitorizações, observações, amostras e multimédia).

  C115 - O utilizador edita uma visita.

  C116 - O utilizador elimina uma visita.

- **Monitorizações**

  C117 - O utilizador consulta as monitorizações realizadas numa visita.

  C118 - O utilizador cria uma monitorização.

  C119 - O utilizador edita uma monitorização.

  C120 - O utilizador elimina uma monitorização.

- **Monitorizações de Agentes**

  C121 - O utilizador consulta as monitorizações realizadas por um agente.

  C122 - O utilizador consulta os agentes que realizaram uma monitorização.

  C123 - O utilizador cria a associação entre uma monitorização e um agente.

  C124 - O utilizador edita a associação entre uma monitorização e um agente.

  C125 - O utilizador elimina a associação entre uma monitorização e um agente.

- **Observações**

  C126 - O utilizador consulta as observações realizadas numa monitorização.

  C127 - O utilizador cria uma observação.

  C128 - O utilizador edita uma observação.

  C129 - O utilizador elimina uma observação.

- **Tipos de Observações**

  C130 - O utilizador consulta os tipos de observações.

  C131 - O utilizador cria um tipo de observação.

  C132 - O utilizador edita um tipo de observação.

  C133 - O utilizador elimina um tipo de observação.

- **Protocolos**

  C134 - O utilizador consulta os protocolos.

  C135 - O utilizador cria um protocolo.

C136 - O utilizador edita um protocolo.

C137 - O utilizador elimina um protocolo.

- **Protocolos em Regiões**

  C138 - O utilizador consulta os protocolos executados numa região em período.

  C139 - O utilizador consulta as regiões em períodos onde um protocolo foi executado.

  C140 - O utilizador cria a associação entre um protocolo e uma região em período.

  C141 - O utilizador edita a associação entre um protocolo e uma região em período.

  C142 - O utilizador elimina a associação entre um protocolo e uma região em período.

- **Observações de Protocolos**

  C143 - O utilizador consulta os tipos de observação de um protocolo.

  C144 - O utilizador consulta os protocolos que utilizam um tipo de observação.

  C145 - O utilizador cria a associação entre um tipo de observação e um protocolo.

  C146 - O utilizador edita a associação entre um tipo de observação e um protocolo.

  C147 - O utilizador elimina a associação entre um tipo de observação e um protocolo.

- **Amostras**

  C148 - O utilizador consulta as amostrar de uma observação.

  C149 - O utilizador cria uma amostra.

  C150 - O utilizador edita uma amostra.

  C151 - O utilizador elimina uma amostra.

- **Tipos de Amostras**

  C152 - O utilizador consulta os tipos de amostras.

  C153 - O utilizador cria um tipo de amostra.

  C154 - O utilizador edita um tipo de amostra.

  C155 - O utilizador elimina um tipo de amostra.

- **Eventos**

  C156 - O utilizador consulta os eventos de uma região em período.

  C157 - O utilizador cria um evento.

C158 - O utilizador edita um evento.

C159 - O utilizador elimina um evento.

- **Tipos de Eventos**

  C160 - O utilizador consulta os tipos de eventos.

  C161 - O utilizador cria um tipo de evento.

  C162 - O utilizador edita um tipo de evento.

  C163 - O utilizador elimina um tipo de evento.

- **Consequências de Eventos**

  C164 - O utilizador consulta as consequências de um evento.

  C165 - O utilizador cria uma consequência de um evento.

  C166 - O utilizador edita uma consequência de um evento.

  C167 - O utilizador elimina uma consequência de um evento.

- **Multimédia**

  C168 - O utilizador consulta o conteúdo multimédia.

  C169 - O utilizador cria conteúdo multimédia.

  C170 - O utilizador edita conteúdo multimédia.

  C171 - O utilizador elimina conteúdo multimédia.

- **Tipos de Multimédia**

  C172 - O utilizador consulta os tipos de conteúdos multimédia.

  C173 - O utilizador consegue cria um tipo de conteúdo multimédia.

  C174 - O utilizador consegue edita um tipo de conteúdo multimédia.

  C175 - O utilizador consegue elimina um tipo de conteúdo multimédia.

- **Fornecedores de Dados**

  C176 - O utilizador consulta os fornecedores de dados.

  C177 - O utilizador cria um fornecedor de dados.

  C178 - O utilizador edita um fornecedor de dados.

  C179 - O utilizador elimina um fornecedor de dados.

- **Tipos de Fornecedores de Dados**

  C180 - O utilizador consulta todos os tipos de fornecedores de dados.

  C181 - O utilizador cria um tipo de fornecedor de dados.

  C182 - O utilizador edita um tipo de fornecedor de dados.

  C183 - O utilizador elimina um tipo de fornecedor de dados.

- **Fontes de Dados**

  C184 - O utilizador consulta as fontes de dados.

  C185 - O utilizador cria uma fonte de dados.

  C186 - O utilizador edita uma fonte de dados.

  C187 - O utilizador elimina uma fonte de dados.

  C188 - O utilizador consulta os dados das séries temporais de uma fonte de dados.

- **Tipo de Fontes de Dados**

  C189 - O utilizador consultar os tipos de fontes de dados.

  C190 - O utilizador cria um tipo de fonte de dados.

  C191 - O utilizador edita um tipo de fonte de dados.

  C192 - O utilizador elimina um tipo de fonte de dados.

- **Dependências de Fontes de Dados**

  C193 - O utilizador consulta as fontes de dados de que uma fonte de dados depende.

  C194 - O utilizador cria a dependência entre duas fontes de dados.

  C195 - O utilizador edita a dependência entre duas fontes de dados.

  C196 - O utilizador elimina a dependência entre duas fontes de dados.

- **Fontes de Dados de Regiões**

  C197 - O utilizador consulta as fontes de dados principais de uma região.

  C198 - O utilizador consulta as regiões que se servem de uma fonte de dados.

  C199 - O utilizador cria a associação entre uma fonte de dados e uma região.

  C200 - O utilizador edita a associação entre uma fonte de dados e uma região.

  C201 - O utilizador elimina a associação entre uma fonte de dados e uma região.

- **Séries**

  C202 - O utilizador consulta as séries de uma fonte de dados.

  C203 - O utilizador cria uma série.

  C204 - O utilizador edita uma série.

  C205 - O utilizador elimina uma série.

- **Tipos de Séries**

  C206 - O utilizador consulta os tipos de séries.

  C207 - O utilizador cria um tipo de série.

C208 - O utilizador edita um tipo de série.

C209 - O utilizador elimina um tipo de série.

- **Derivação de Séries**

  C210 - O utilizador consulta as séries que derivam uma série.

  C211 - O utilizador cria a relação de derivação entre duas séries.

  C212 - O utilizador edita a relação de derivação entre duas séries.

  C213 - O utilizador elimina a relação de derivação entre duas séries.

- **Identificadores de Séries**

  C214 - O utilizador consulta os identificadores de séries.

  C215 - O utilizador cria um identificador de série.

  C216 - O utilizador edita um identificador de série.

  C217 - O utilizador elimina um identificador de série.

- **Métodos de Cálculo**

  C218 - O utilizador consulta os métodos de cálculo.

  C219 - O utilizador cria um método de cálculo.

  C220 - O utilizador edita um método de cálculo.

  C221 - O utilizador elimina um método de cálculo.

- **Períodos de Séries**

  C222 - O utilizador consulta os períodos de séries.

  C223 - O utilizador cria um período de série.

  C224 - O utilizador edita um período de série.

  C225 - O utilizador elimina um período de série.

- **Tipos de Dados**

  C226 - O utilizador consulta os tipos de dados.

  C227 - O utilizador cria um tipo de dados.

  C228 - O utilizador edita um tipo de dados.

  C229 - O utilizador elimina um tipo de dados.

- **Unidades**

  C230 - O utilizador consulta as unidades.

  C231 - O utilizador cria uma unidade.

  C232 - O utilizador edita uma unidade.

  C233 - O utilizador elimina uma unidade.

APÊNDICE C

Visualizações dos Dados

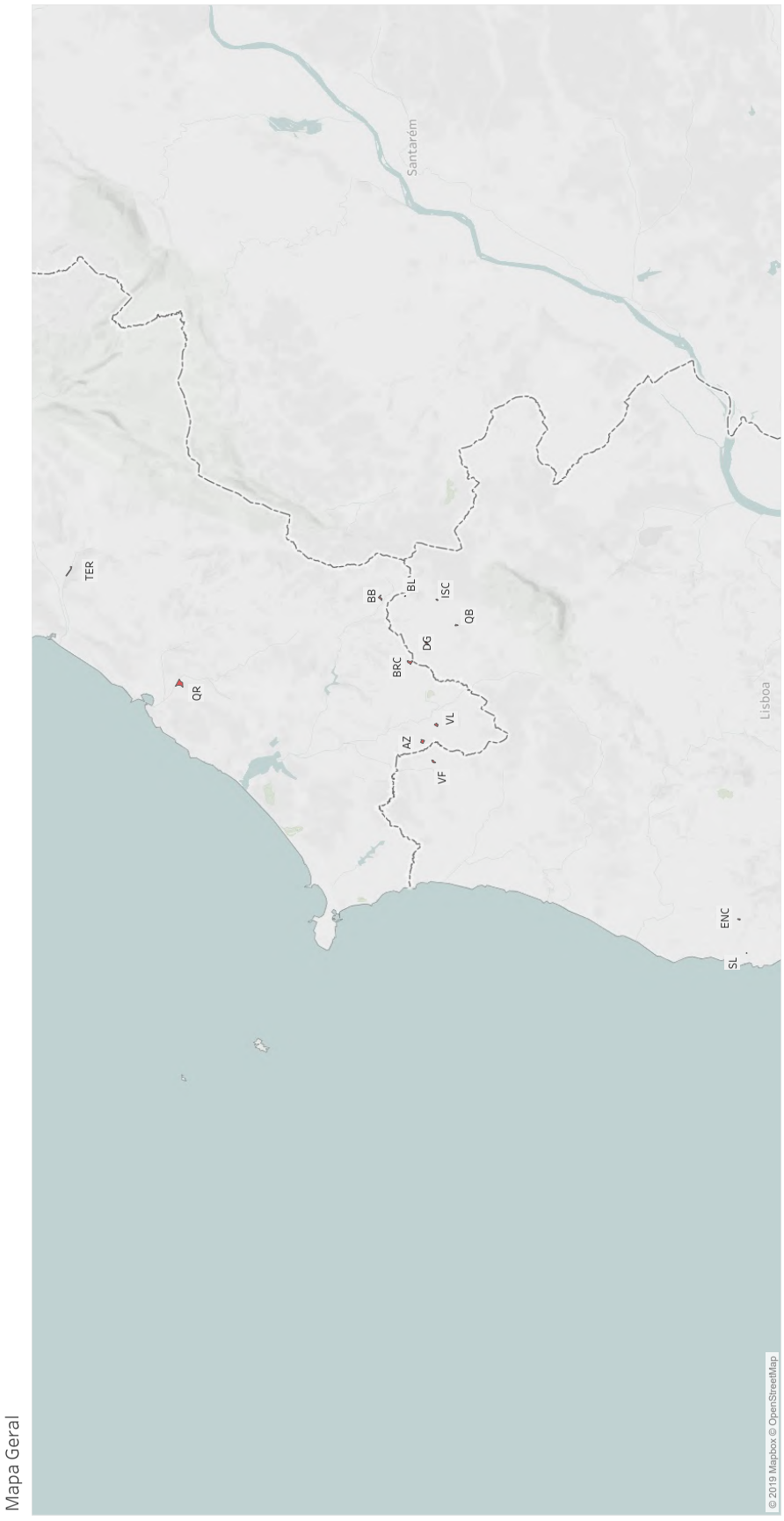


Figura C.1: Mapa Geral

Figura C.2: Elementos de Interesse de Parcela

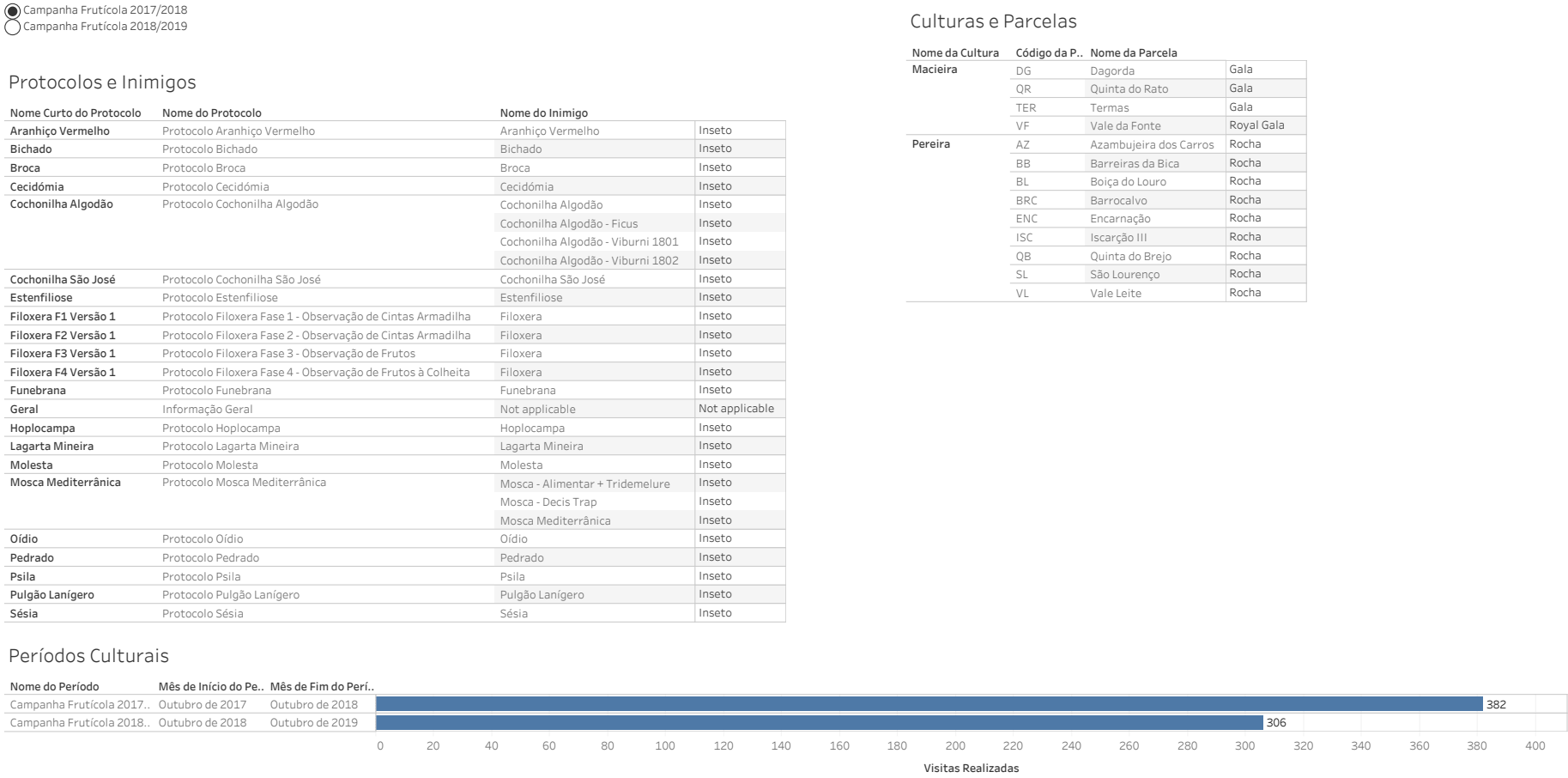


Figura C.3: Parcelas em Observação, Protocolos em Execução e Número Total de Visitas

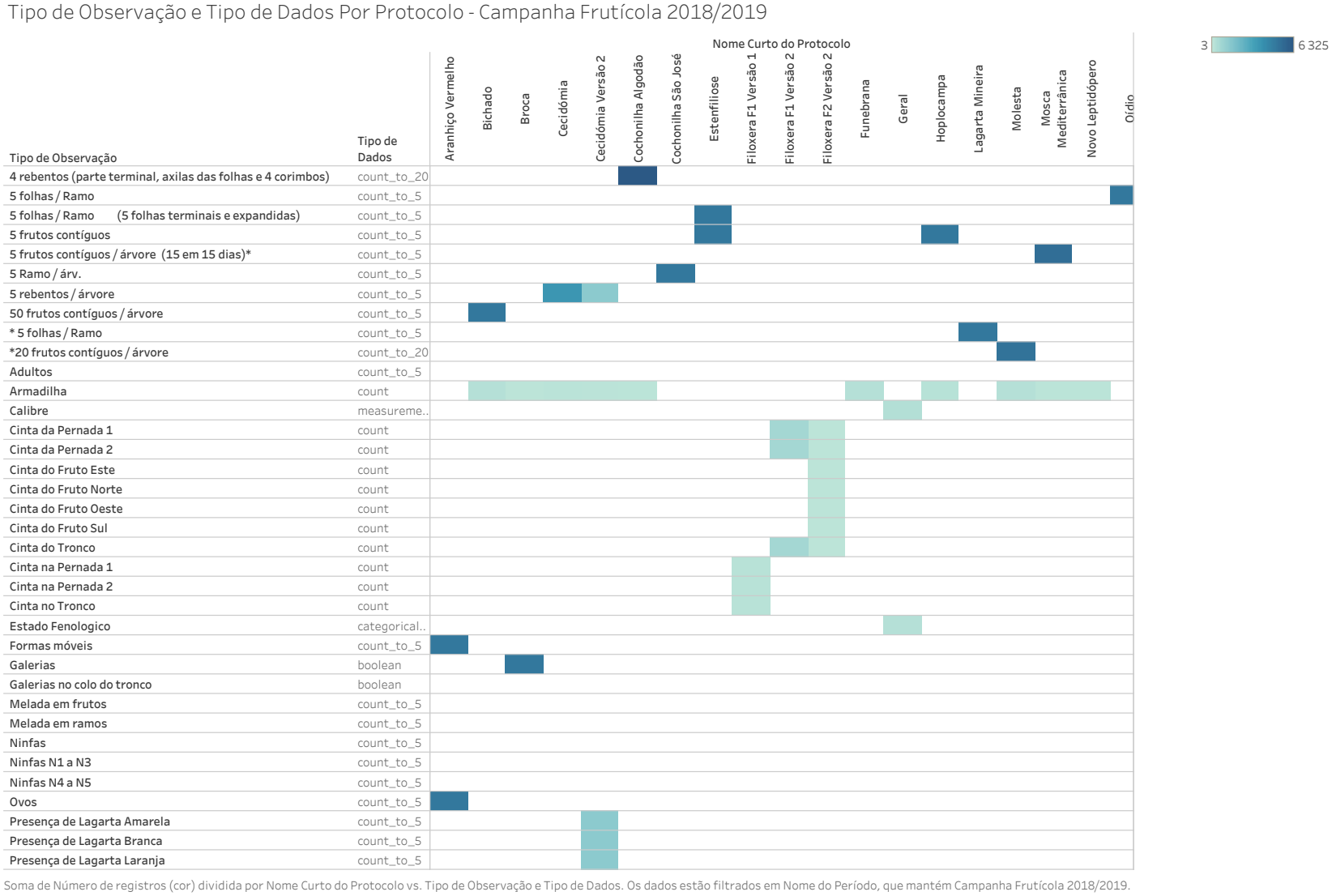


Figura C.4: Tipo de Observação e Tipos de Dados por Protocolo

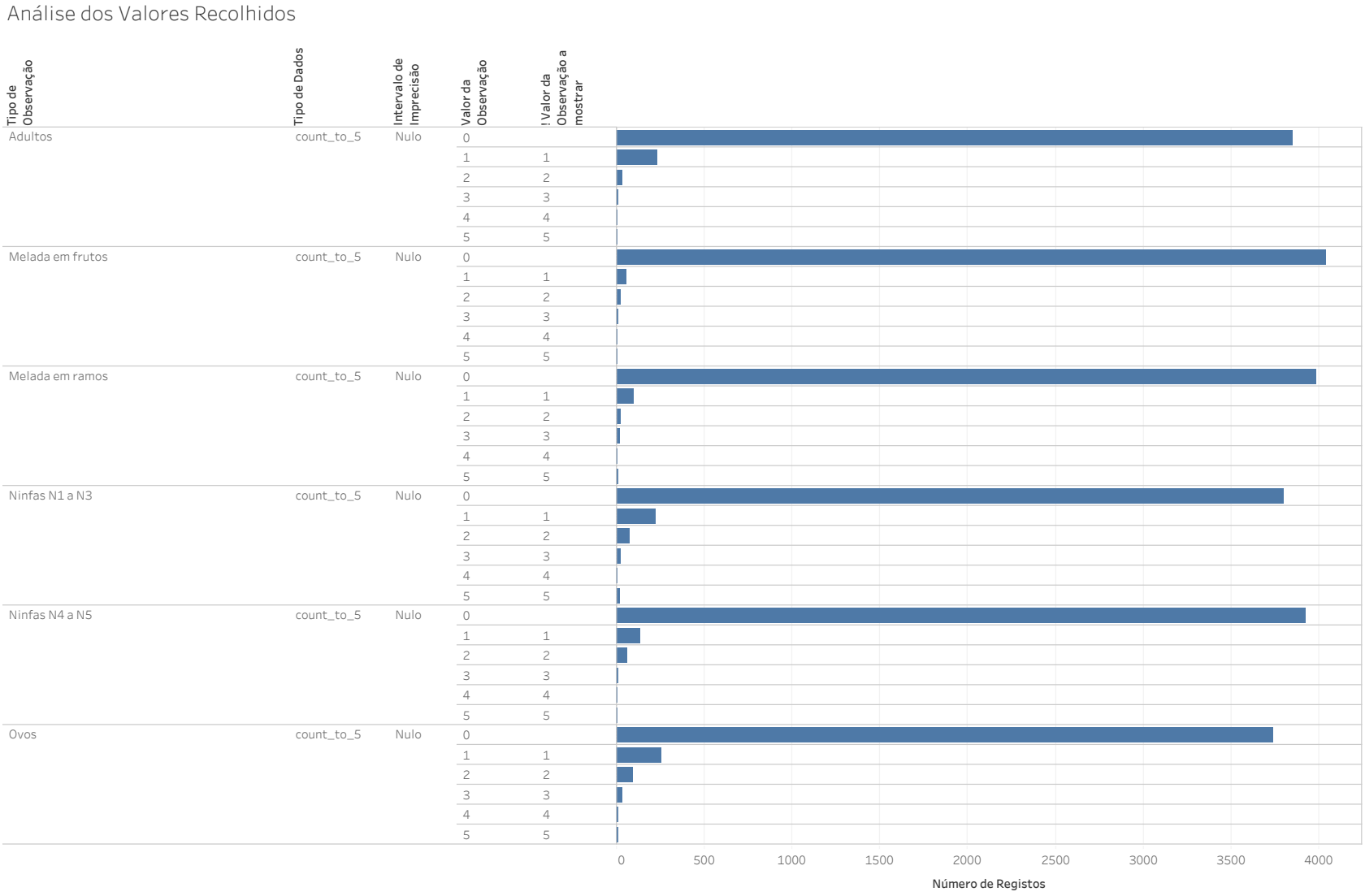


Figura C.5: Análise dos Valores Recolhidos

Visitas por Protocolos x Parcelas - Campanha Frutícola 2018/2019

Código da Parcela

2 29

Nome Curto do Protoc..	AZ	BB	BL	BRC	DG	ENC	ISC	QB	QR	SL	TER	VF
Aranhiço Vermelho	18	22	16	20	22	16	16	22	16	22	21	21
Bichado	18	22	16	20	22	16	16	22	17	22	21	21
Broca	18	22	16	20	22	16	16	22	16	22	21	21
Cecidómia	2	22		20	22	16	16	6	16	6	21	21
Cecidómia Versão 2	16		16					16		16		
Cochonilha Algodão	18	22	16	20	22	16	16	22	16	22	21	21
Cochonilha São José	18	22	16	20	22	16	16	22	16	22	21	21
Estenfiliose	18	22	16	20	22	16	16	22	16	22	21	21
Filoxera F1 Versão 1	3			3		3	4					
Filoxera F1 Versão 2	4			16		29	17					
Filoxera F2 Versão 2						3						
Funebrana	14	14	15	14	17	15	15	14	16	16	13	15
Geral	18	22	16	20	22	16	16	22	16	22	21	21
Hoplocampa	18	22	16	20	22	16	16	22	16	22	21	21
Lagarta Mineira	18	22	16	20	22	16	16	22	16	22	21	21
Molesta	18	22	16	20	22	16	16	22	17	22	21	21
Mosca Mediterrânica	20	22	16	21	22	16	16	22	16	22	21	21
Novo Leptidópero					14				9		13	
Oídio	18	22	16	20	22	16	16	22	16	22	21	21
Pedrado	18	22	16	20	22	16	16	22	16	22	21	21
Psila	2	6		5	6			6		6	5	5
Psila Versão 2	16	16	16	15	16	16	16	16	16	16	16	16
Pulgão Lanígero	18	22	16	20	22	16	16	22	16	22	21	21
Sésia	18	22	16	20	22	16	16	22	16	22	21	21
Total geral	25	22	16	25	22	38	34	22	17	22	21	21

Contagem distinta de Idvisita dividido por Código da Parcela vs. Nome Curto do Protocolo. A cor mostra contagem distinta de Idvisita. As marcas são rotuladas por contagem distinta de Idvisita. Os dados estão filtrados em Nome do Período, que mantém Campanha Frutícola 2018/2019.

Figura C.6: Visitas por Protocolos x Parcelas

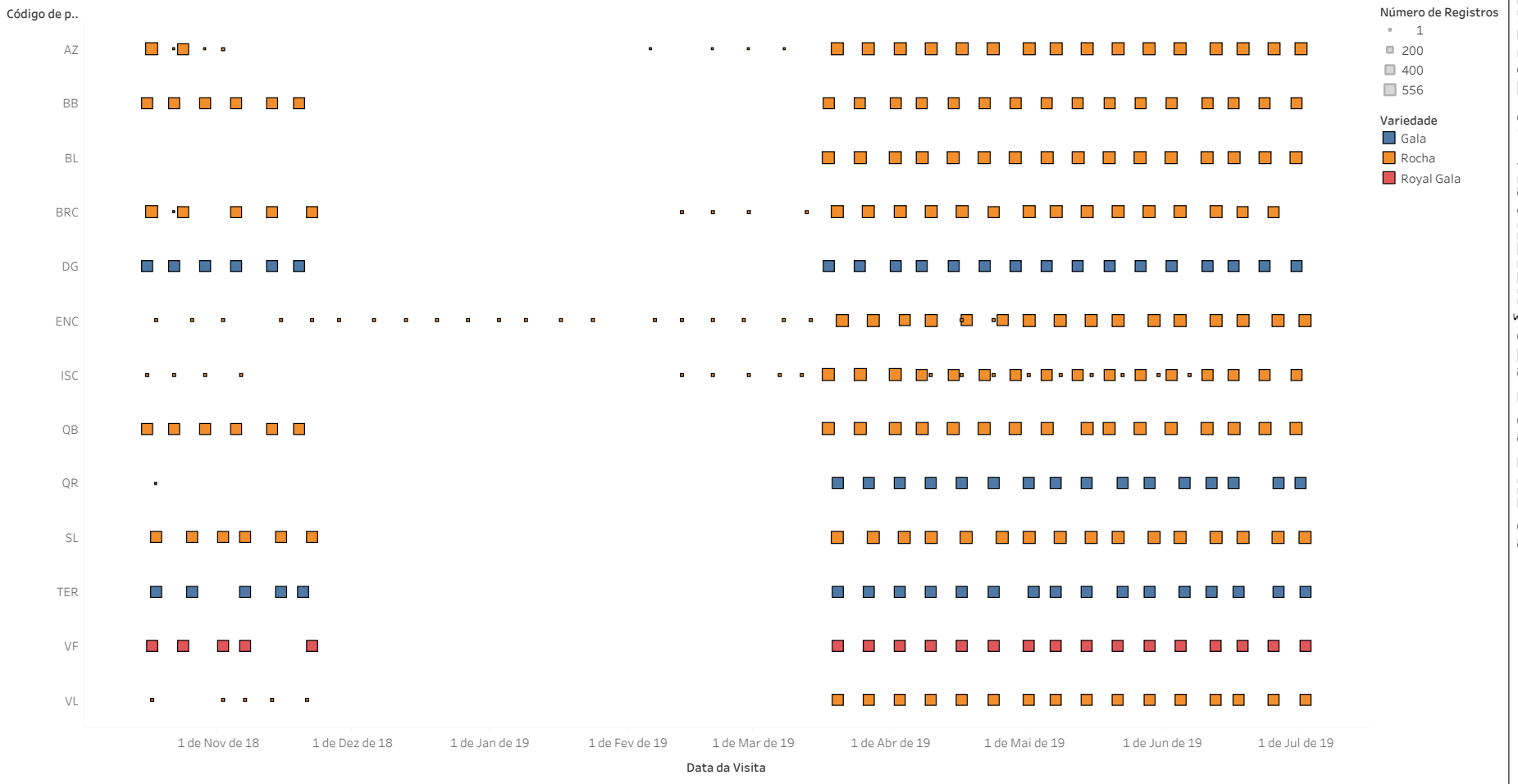


Figura C.7: Visitas nas Parcelas

28 de Março de 2019

Protocolo	Tipo de observação	Nulo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
		Nomeeoi																									
Aranhiço Vermelho	Formas móveis																										
	Ovos																										
Bichado	50 frutos contíguos / árvore																										
	Armadilha																										
Broca	Galerias																										
Cecidómia	5 rebentos / árvore																										
	Armadilha																										
Cochonilha Algodão	4 rebentos (parte terminal, axil..																										
Cochonilha São José	5 Ramo / árv.																										
Estenfiliose	5 folhas / Ramo (5 folhas ter..																										
	5 frutos contíguos																										
Filoxera F1 Versão 2	Cinta da Pernada 1							> 20	> 20	> 20	2	> 20															
	Cinta da Pernada 2									13	> 20																
	Cinta do Tronco								4	14		11															
Funebrana	Armadilha	1																									
Hoplocampa	5 frutos contíguos																										
	Armadilha																										
Lagarta Mineira	* 5 folhas / Ramo																										
Molesta	*20 frutos contíguos / árvore																										
	Armadilha	1																									
Mosca Mediterrânica	5 frutos contíguos / árvore (15 ..																										
N/A	Calibre	N/A																									
	Estado Fenologico	D3/E																									
Oídio	5 folhas / Ramo																										
Pedrado	5 folhas / Ramo (5 folhas ter..																										
	5 frutos contíguos																										
Psila Versão 2	Adultos						1						1														
	Melada em frutos																										
	Melada em ramos																										
	Ninfas N1 a N3				1				2					1		4					1						
	Ninfas N4 a N5																										
	Ovos																										
Pulgão Lanígero	5 Ramo / árv.																										
Sésia	Galerias no colo do tronco																										

Figura C.8: Detalhes de uma Visita

Status das Fontes

Fornecedor	Tipo de Fonte	Fonte de Dados	Primeira Da..	Última Data	
APAS	Estação Meteorológica	Apas04Fr	15/03/2018	06/09/2019	
		Apas05Do	15/11/2017	06/09/2019	
		CABAzamb	18/04/2018	06/09/2019	
		CABValeC	17/04/2018	06/09/2019	
		Coopval01VV	13/04/2018	06/09/2019	
		Coopval02AL	13/04/2018	06/09/2019	
		Coopval03VN	19/04/2018	06/09/2019	
		CPFAbelh	04/05/2018	06/09/2019	
		CPFAdosF	03/05/2018	06/09/2019	
		CPFQuint	20/04/2018	06/09/2019	
		CPFSanta	24/04/2018	06/09/2019	
		FrBarrei	08/05/2018	06/09/2019	
		FrEricei	28/05/2018	16/07/2019	
		FrTurcif	08/05/2018	06/09/2019	
IPMA	Estação Meteorológica	Alcobaça	30/06/2019	06/09/2019	
		Cabo Carvoeiro	30/06/2019	06/09/2019	
		Leiria (Aeródromo)	30/06/2019	06/09/2019	
		Rio Maior	30/06/2019	06/09/2019	
		Santa Cruz (Aerodromo)	30/06/2019	06/09/2019	
		Santarém, Fonte Boa	30/06/2019	06/09/2019	
		Torres Vedras, Dois Portos	30/06/2019	06/09/2019	
	Previsão Meteorológica	Alcobaça	29/07/2019	15/09/2019	
		Almeirim	29/07/2019	15/09/2019	
		Alpiarça	30/07/2019	15/09/2019	
		Bombarral	29/07/2019	15/09/2019	
		Cadaval	29/07/2019	15/09/2019	
		Caldas da Rainha	29/07/2019	15/09/2019	
		Cartaxo	29/07/2019	15/09/2019	
		Ericeira	29/07/2019	15/09/2019	
		Lourinhã	29/07/2019	15/09/2019	
		Mafra	29/07/2019	15/09/2019	
		Nazaré	29/07/2019	15/09/2019	
		Óbidos	29/07/2019	15/09/2019	
		Praia de Coxos	29/07/2019	15/09/2019	
		Praia de Santa Cruz	29/07/2019	15/09/2019	
		Praia de São Martinho do ..	29/07/2019	15/09/2019	
		Rio Maior	29/07/2019	15/09/2019	
		Santarém	29/07/2019	15/09/2019	
		Sobral de Monte Agraço	29/07/2019	15/09/2019	
		Torres Vedras	29/07/2019	15/09/2019	
PESSL	Estação Meteorológica	AARA Alfeizerao	23/01/2013	10/07/2019	
		AARA Cela	01/01/2013	06/09/2019	
		AARA Maiorga	01/01/2013	09/08/2019	
		AARA Montes	15/05/2013	06/09/2019	
		Abadias	11/04/2018	06/09/2019	
		COTHN	01/01/2013	06/09/2019	
		Évora de Alcobaça	11/04/2018	04/09/2019	
		Quinta do Matao	22/04/2013	06/09/2019	
		Quinta Nova (Alcobaça)	01/01/2013	06/09/2019	
		Santa Catarina	01/01/2013	06/09/2019	
TerraPro	Estação Meteorológica	Carrascal	11/05/2019	30/05/2019	
		Runa	11/05/2019	30/05/2019	
		Silveira	11/05/2019	30/05/2019	
WiseCrop	Estação	Iscarção	20/03/2018	06/09/2019	

Status
Ativa
Atrasada
Offline

Status (cor) dividida por Fornecedor, Tipo de Fonte, Fonte de Dados, Primeira Data e Última Data

Figura C.9: Status das Fontes de Dados